Lignes d'usinage avec équipements standard : modélisation, configuration et optimisation

Sana Belmokhtar

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 424 GI

THÈSE

présentée par

Sana Belmokhtar

pour obtenir le grade de

Docteur de l'École Nationale Supérieure
des Mines de Saint-Etienne

spécialité génie industriel

Lignes d'usinage avec équipements standard : modélisation, configuration et optimisation

Soutenue le 11 Décembre 2006
en présence d'un jury composé de :

Michel ALDANONDO	Professeur, École des Mines d'Albi	président
Fayez BOCTOR	Professeur, Université Laval, Québec	rapporteur
Michel GOURGAND	Professeur, Université de Clermont Ferrand	rapporteur
Aziz MOUKRIM	Professeur, Université de Technologie de Compiègne	rapporteur
El-Houssaine AGHEZZAF	Professeur, Université de Gent, Belgique	examinateur
Alexandre DOLGUI	Professeur, École des Mines de St Etienne	directeur
Xavier DELORME	Maître-Assistant, École des Mines de St Etienne	co-directeur

•

Spécialités doctorales :	**Responsable :**
SCIENCES ET GENIE DES MATERIAUX	**J. DRIVER** Directeur de recherche - Centre SMS
MECANIQUE ET INGENIERIE	**A. VAUTRIN** Professeur - Centre SMS
GENIE DES PROCEDES	**G. THOMAS** Professeur - Centre SPIN
SCIENCES DE LA TERRE	**B. GUY** Maître de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	**J. BOURGOIS** Professeur - Centre SITE
MATHEMATIQUES APPLIQUEES	**E. TOUBOUL** Ingénieur
INFORMATIQUE	**O. BOISSIER** Professeur - Centre G2I
IMAGE, VISION, SIGNAL	**JC. PINOLI** Professeur - Centre CIS
GENIE INDUSTRIEL	**P. BURLAT** Professeur - Centre G2I
MICROELECTRONIQUE	**Ph. COLLOT** Professeur - Centre CMP

• Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'Etat ou d'une HDR)

BENABEN	Patrick	PR 2	Sciences & Génie des Matériaux	SMS
BERNACHE-ASSOLANT	Didier	PR 1	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	MR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 2	Informatique	G2I
BOUDAREL	Marie-Reine	MA	Sciences de l'inform. & com.	DF
BOURGOIS	Jacques	PR 1	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	MR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 1	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	ICM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 2	Sciences & Génie de l'Environnement	SITE
DELAFOSSE	David	PR 2	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Informatique	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	CIS
DRIVER	Julian	DR	Sciences & Génie des Matériaux	SMS
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	SMS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	CMP
FRACZKIEWICZ	Anna	MR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	CR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GOEURIOT	Patrice	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUILHOT	Bernard	DR	Génie des Procédés	CIS
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
JOYE	Marc	Ing. (Gemplus)	Microélectronique	CMP
KLÖCKER	Helmut	CR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LE COZE	Jean	PR 1	Sciences & Génie des Matériaux	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MOLIMARD	Jérôme	MA	Sciences & Génie des Matériaux	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	MA1	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 1	Image, Vision, Signal	CIS
SOUSTELLE	Michel	PR 1	Génie des Procédés	SPIN
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
THOMAS	Gérard	PR 1	Génie des Procédés	SPIN
TRAN MINH	Cahn	MR	Génie des Procédés	SPIN
VALDIVIESO	Françoise	CR	Génie des Procédés	SPIN
VAUTRIN	Alain	PR 1	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	CR	Génie des procédés	SPIN
WOLSKI	Krzysztof	CR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR 1	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
ICM	Ingénieur en chef des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
SITE	Sciences Information et Technologies pour l'Environnement
G2I	Génie Industriel et Informatique
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Remerciements

Cette thèse n'aurait pu voir le jour sans l'aide et le soutien de nombreuses personnes, aussi je voudrais simplement leur exprimer ici toute ma reconnaissance et ma gratitude.

Je voudrais tout d'abord remercier mon directeur de thèse Alexandre Dolgui pour m'avoir donné l'opportunité de travailler avec lui ; je tiens particulièrement à lui témoigner ma sincère reconnaissance pour tout ce qu'il m'a apporté durant ces trois dernières années, pour la bonne ambiance qu'il a su créer, sa continuelle disponibilité et ses précieux conseils.

Mes remerciements vont à mes rapporteurs qui m'ont fait l'honneur d'accepter d'évaluer mon travail. Je remercie Fouad Fayez Boctor professeur de l'université LAVAL pour ses critiques et pour l'intérêt particulier qu'il a porté à mes travaux. Je remercie Aziz Moukrim professeur à l'Université de Technologie de Compiègne de m'avoir fait l'honneur et le plaisir de rapporter ma thèse ; je lui suis très reconnaissante pour tous ses encouragements et pour ses conseils. Je remercie également Michel Gourgand professeur de l'université de Clermont-Ferrand d'avoir accepté de rapporter ma thèse. J'ai été très honorée par ses appréciations. Enfin, je remercie le professeur Michel Aldanondo d'avoir accepté de présider ce jury et le professeur El-Houssaine Aggezzaf de m'avoir fait l'honneur d'examiner mon travail.

Je voudrais remercier également Xavier Delorme d'avoir bien voulu intégrer la co-direction de ma thèse. Je lui suis particulièrement reconnaissante pour toutes les discussions que nous avons pu avoir durant ces deux dernières années et pour ses remarques toujours pertinentes qui ont permis d'améliorer la qualité de mes travaux. Je ne peux pas oublier de remercier Hélène Marian et Frédéric Grimaud qui m'ont aussi bien accueillie et intégrée au sein de l'équipe.

Un grand merci à celle et ceux et avec qui j'ai eu à travailler pendant ma thèse, je cite Antoneta Bratcu, Levin Genrikh, Nikolai Guschinsky et Ivan Inhatsenka sans eux ma thèse ne saurait être ce qu'elle est aujourd'hui.

Enfin, mille et un mercis à tous ceux et celles qui, par un mot, un sourire ou un geste ont fait de mon passage à Saint-Etienne un moment inoubliable. Enfin, MERCI aux êtres qui me sont chers : mes amis, ma grande famille, surtout mes parents sans oublier ma sœur et mes deux frères, grâce à eux j'ai pu garder le cap et arriver à bon port.

Table des matières

Introduction générale

Durant ces dernières années, nous avons assisté à l'établissement d'une nouvelle conjoncture économique engendrée par une mondialisation accrue imposant des conditions de compétitivité à l'échelle du globe. Ces circonstances réduisent de manière significative les marges d'erreur des décideurs et leur imposent une réactivité et une efficacité absolues dans le seul but de conserver des parts de marché. L'enjeu économique est d'envergure et les décideurs ne peuvent plus s'appuyer sur leur intuition ou même un savoir-faire issu d'une longue expérience.

C'est dans ce contexte que s'inscrivent nos contributions en développant des outils d'aide à la décision pour la configuration des lignes d'usinage. Les approches qui nous intéressent permettent une représentation formelle des problèmes en se basant sur une modélisation en programmation mathématique, plus particulièrement en programmation linéaire en nombres entiers (PLNE). Ce choix a été motivé par les évolutions incontestables qu'a connues cette branche de la programmation mathématique en offrant des outils de plus en plus performants et accessibles tels que ILOG Cplex et Dash Xpress MP, qui sont probablement les plus connus.

D'autre part, la programmation logique par contraintes (PPC) qui est issue de l'intelligence artificielle a suscité un intérêt grandissant pour la résolution des problèmes complexes. Cette méthode a la propriété de faire participer activement les contraintes dans le processus de résolution ; de plus, comme pour la programmation linéaire en nombres entiers, des outils spécifiques ont été développés dans ce cadre (ILOG Solver, par exemple). L'ensemble de ces facteurs nous a amené à penser que cette approche pourrait également offrir un cadre adéquat pour l'optimisation du problème traité dans ce mémoire.

Nous avons défini un problème théorique nouveau en partant de problèmes industriels concrets pour l'étude des lignes d'usinage équipées d'unités standard. Pour ces lignes, nous distinguons plusieurs modes d'activation pour les unités appartenant à une même station. En effet, nous relevons le mode parallèle où toutes les unités d'usinage sont enclenchées en même temps marquant ainsi le début d'un cycle. Dans le mode séquentiel, les unités sont activées les unes après les autres au sein d'une station. Enfin, le mode mixte est une généralisation des deux modes précédents. Dans ce mode, une station est composée de plusieurs étages séquentiels tels que chacun de ces étages peut contenir différentes unités qui sont, quant à elles, activées en parallèle.

Le problème qui nous intéresse dans cette thèse, est posé comme la définition de la meilleure configuration en termes de choix d'équipement (unités d'usinage) pour les stations à partir d'un catalogue défini au préalable. En pratique, ce catalogue est construit suite à

une étude de disponibilité d'unités d'usinage capables de répondre aux besoins afin d'effectuer l'ensemble des opérations définies dans la gamme d'usinage. Cette phase de collecte d'information est normalement réalisée dès que l'ensemble des gammes est déterminé. Une fois cet ensemble connu, le problème revient à déterminer un sous-ensemble d'unités à partir de cet ensemble, tel que toutes les opérations nécessaires à la fabrication du produit soient effectuées. Les opérations des unités sélectionnées doivent correspondre exactement aux opérations nécessaires à la fabrication du produit. L'affectation des unités aux stations doit être réalisée de manière à ce que toutes les contraintes soient vérifiées en minimisant le coût total constitué du coût des unités équipant les stations ainsi que le coût des stations elles-mêmes.

La recherche de la meilleure structure minimisant le coût d'investissement est une question prépondérante lors de la conception des lignes d'usinage. Le coût de la mise en place d'un tel système est important atteignant quelques dizaines de millions d'euros. Par conséquent, le choix d'une bonne configuration peut permettre un gain considérable. Lorsqu'un tel système est inexistant et qu'il faut le mettre en place, sa conception préliminaire constitue l'étape la plus importante, car toute modification postérieure à cette étape serait extrêmement coûteuse. Pour les lignes déjà en fonctionnement, les reconfigurations sont également à étudier avec minutie ; dans ce cas, le critère peut être différent du coût d'acquisition, le temps et le coût de reconfiguration semblent, par exemple, plus appropriés. Toutefois, qu'il s'agisse de conception préliminaire ou de reconfiguration il faut définir une structure de la ligne en précisant les unités retenues pour équiper les stations en respectant les contraintes liées aux opérations, aux unités d'usinage et à la productivité de la ligne.

Ce mémoire est structuré en trois parties, chacune comportant un ou plusieurs chapitres. La première partie de ce mémoire est composée des trois premiers chapitres que nous décrivons comme suit :

Le **chapitre 1** introduit le contexte général de notre étude en décrivant les systèmes de production dans leur ensemble puis les lignes d'usinage de façon plus particulière. Nous décrivons les différents types de lignes qui existent en soulignant, à chaque fois, leurs particularités. Enfin, nous abordons le problème traité dans ce mémoire, noté TLBP/B-P, en décrivant les données et différentes contraintes considérées.

Dans le **chapitre 2**, nous rappelons quelques définitions liées au domaine de la recherche opérationnelle qui sont indispensables à la compréhension de la suite du mémoire. Nous y soulignons particulièrement les méthodes employées, à savoir : la programmation linéaire en nombres entiers et la programmation par contraintes.

Dans le **chapitre 3**, nous passons en revue les travaux de la littérature que nous rapportons. Nous reprenons les travaux effectués pour les problèmes d'équilibrage des lignes d'assemblage (SALBP) en raison des nombreux points de similitudes qu'ils comportent avec le problème étudié dans ce mémoire. Nous terminons cette première partie par les problèmes d'équilibrage liés aux lignes d'usinage.

La seconde partie (chapitre 4, 5 et 6) constitue le cœur de ce mémoire. Dans le **chapitre 4**, nous présentons formellement le problème de configuration qui nous intéresse en apportant un exemple industriel concret illustrant l'ensemble des données et contraintes. Nous fournissons également plusieurs règles de pré-traitement qui ont pour objectif de ré-

duire la taille des données d'entrée. Le **chapitre 5** est réservé à la présentation des différents modèles proposés. Nous suggérons plusieurs algorithmes pour réduire la taille des modèles. Enfin dans le **chapitre 6**, nous rapportons les résultats de la phase expérimentale qui a été menée. L'objectif de cette phase est triple : d'abord évaluer les performances de chacun des modèles proposés puis étudier l'effet des algorithmes de réduction et enfin mesurer l'influence de certaines caractéristiques numériques du problème.

La dernière partie du mémoire (chapitre 7 et 8) est consacrée à quelques extensions du problème traité. Nous y décrivons les travaux en cours ainsi que les possibilités qui s'ouvrent pour les travaux futurs. Plus précisément, nous abordons dans le **chapitre 7**, une première généralisation, du point de vue du mode d'activation des unités d'usinage au sein des stations, en considérant le mode d'activation mixte. Dans ce cas, les stations sont composées d'étages séquentiels, chacun d'eux pouvant contenir plusieurs unités d'usinage. Pour le cas mixte, il faut introduire de nouvelles contraintes concernant l'incompatibilité des unités d'un même étage ainsi que des contraintes explicites pour le respect du temps de cycle.

Dans le **chapitre 8**, nous présentons une seconde généralisation, cette fois-ci, du point de vue du nombre de produits fabriqués sur la ligne. Nous montrons comment le problème de conception des lignes multi-produit peut être résolu à partir du modèle linéaire proposé initialement pour le cas mono-produit. Pour ce faire, nous suggérons une agrégation des contraintes et données pour l'ensemble des produits considérés ainsi que des règles de maintien de cohérence à appliquer en tant que pré-traitements.

Première partie

État de l'art

Chapitre 1

Optimisation des lignes d'usinage

Au fil du temps, la production[1] a évolué au rythme des époques. Elle est passée de l'ère de l'industrie primaire qui se réduisait alors à l'agriculture, la pêche, la chasse et l'exploitation minière, à l'industrie secondaire qui s'est développée essentiellement autour de produits tangibles. Le début du XIX^{ème} siècle a été marqué par les pionniers de l'économie tels que Adams Smith (1723-90) et John Stuart Mill (1806-73) qui ont introduit la notion de manufacture[2] et le concept de commercialisation des produits sur le marché. Enfin, la dernière phase a débuté à la fin du XIX^{ème} siècle pendant laquelle l'industrie tertiaire a élargi le concept de production à la création d'utilités au sens de services [Hit96].

Dans ce qui suit, nous introduisons les systèmes de production d'une façon générale en portant un intérêt particulier aux systèmes d'usinage car se sont ces derniers qui font l'objet de l'étude menée dans cette thèse. Ainsi, nous rapportons d'abord la terminologie retenue dans ce mémoire puis nous présentons de manière plus détaillée les différents types de systèmes auxquels nous nous sommes intéressés.

1.1 Description des systèmes de production

Plusieurs définitions sont attribuées à la production, nous en rapportons quelques unes que nous citons par ordre chronologique.

D'après Dano [Dan66], la production a été définie comme étant une série de transformations de l'état de matières premières à l'état de produits finis ; chacune de ces transformations correspond à des changements physiques ou chimiques des matières traitées. Askin et Standridge [AS93] proposent la définition suivante : «*la production est le passage d'une conception à un produit fini*». Quant à Hitomi [Hit96], il la définit comme l'action de créer un élément tangible ou non tangible (un service).

Ces définitions englobent l'ensemble des types de systèmes de production : ainsi qu'il s'agisse d'assemblage, de montage ou d'usinage elles restent valables. Toutefois et en dépit

[1] Le terme production est apparu en 1483, il provient du mot latin *producere*.
[2] Le mot manufacture vient du mot latin *manu factum* qui signifie fait à la main.

des propriétés communes que peuvent avoir ces types de systèmes, les systèmes d'usinage ont des caractéristiques qui leur sont propres.

1.1.1 Terminologie générale

Un système de production transforme de la matière première qui lui est fournie en entrée pour en faire un produit fini en sortie. Le système est composé d'un ou de plusieurs postes de travail appelés, plus communément, *stations*. Une station consiste généralement en un groupement de machines ou d'opérateurs humains qui effectuent de façon répétitive les mêmes opérations [HS00].

Les opérations qui sont effectuées dans ces stations peuvent être séparées entre celles effectuées par des opérateurs humains et celles qui sont réalisées par des machines (automatisées). Nous utilisons le terme de *pièce* pour désigner l'entité en cours de transformation dans le système mais également le produit fini qui en résulte. Selon le type de production, une pièce peut être un composant à usiner ou un élément à assembler. De plus, toutes les pièces qui sont dans le système de production, quelque soit leur état, sont considérées comme des *en-cours* [HS00]. Parmi ces derniers, il faut distinguer les *stocks tampons* qui peuvent être mis en place entre les stations selon la politique de gestion retenue. Leur objectif est de couvrir la production pour pallier aux aléas du système tels que les défaillances des machines [DP06].

1.1.2 Mesures de performance

Concernant les mesures de performances d'un système de production, la plus utilisée est sans doute le *taux de productivité* ou productivité (appelé par les anglo-saxons : *throughput*). La productivité d'un système correspond au nombre d'unités produites par unité de temps. À l'inverse, si nous raisonnons par rapport au temps nécessaire à la production d'un produit c'est du *temps de cycle* dont il s'agit. En pratique, il correspond au temps qui sépare la sortie de deux produits du système. La productivité est à l'évidence inversement proportionnelle au temps de cycle. Ainsi, pour augmenter la productivité d'un système il est indispensable de réduire son temps de cycle.

Néanmoins, il est à signaler qu'il existe une autre définition du temps de cycle qui le détermine comme étant le temps que passe la pièce dans le système de production. Ce qui revient à cumuler les temps des postes de travail du système lorsque ceux-ci sont agencés en série. Pour éliminer toute ambiguïté, nous désignons ce dernier par *temps de séjour*.

Une mesure importante de performance est le coût unitaire estimé comme étant le rapport entre le coût de fonctionnement dans une période donnée et le nombre de produits fabriqués dans cette période. Pour un coût global fixe, il est clair que plus la productivité est importante moins le coût unitaire est élevé. De la même manière, pour une productivité fixe, plus le coût global est faible plus le coût unitaire est réduit.

1.1.3 Des principes de base

L'étude du comportement d'un système de production est un problème complexe. Par exemple, si un système comporte K postes de travail et que chacun d'entre eux peut prendre m états[3] alors le système a K^m états potentiels. Ce nombre peut très vite atteindre une très grande valeur, ce qui le rend difficile à étudier [AS93].

Selon la façon dont sont disposés les postes de travail, nous distinguons les lignes sérielles où les postes sont en série, de celles où chaque station en série est dupliquée en des postes en parallèle. Il faut également noter les architectures mixtes où certaines stations sont composées que d'un seul poste de travail alors que d'autres en ont plusieurs en parallèle. Dans nos travaux, nous nous sommes intéressés aux lignes sérielles qui présentent deux avantages majeurs : elles sont simples à gérer et moins onéreuses.

1.2 Les lignes d'usinage

1.2.1 L'usinage

L'usinage tient une place importante dans l'industrie mondiale et tout particulièrement dans l'industrie européenne. Le rapport du centre européen des statistiques [Joh06] permet de le confirmer car la fabrication des machines et équipements représente la troisième division manufacturière en Europe en terme de valeur ajoutée. Celle-ci se chiffre à 165 milliards d'euro, ce qui représente 10,8% de la valeur ajoutée totale de l'industrie manufacturière européenne. Ce rapport permet également de voir que le marché des machines-outils représente plus de 14 milliards d'euros, soit 8,8% de la production des machines et équipements.

L'usinage par enlèvement de copeaux a de nombreuses spécificités influençant directement l'organisation de ces systèmes de production. De ce fait, leur étude nécessite une attention particulière et une prise en compte minutieuse de ces caractéristiques. Dans ce qui suit, nous introduisons les opérations les plus fréquentes dans l'usinage avant d'aborder plus en détail les différents types de lignes d'usinage [Wik06].

- **Le perçage** : cette opération d'usinage permet de faire un trou dans une pièce. Celui-ci peut être de bout en bout ou ne pas déboucher.
- **Le fraisage** : cette opération est réalisée à l'aide de fraises qui permettent d'enlever de la matière en combinant un mouvement double composé d'une rotation et d'un déplacement vertical (ou horizontal).
- **Le taraudage** : un trou taraudé correspond à la forme complémentaire d'une vis ou tige filetée. Techniquement, il s'agit d'un trou lisse dans lequel on opère un filetage.
- **L'alésage** : cette opération, qui permet d'affiner la surface intérieure d'un cylindre, a pour but de calibrer la précision dimensionnelle dictée par des normes industrielles de qualité.

[3] Les états peuvent correspondre à différentes opérations effectuées.

FIG. 1.1 – Quelques pièces fabriquées par PCI Scemm

La combinaison de l'ensemble de ces opérations permet de réaliser des pièces complexes telles que celles que nous rapportons dans la figure 1.1. Ces images nous proviennent de notre partenaire industriel PCI qui est le leader français de la conception, la réalisation et la mise en service de biens d'équipement industriels[4].

1.2.2 Les lignes d'usinage dédiées

La production de masse a connu un grand essor en employant les principes d'interchangeabilité et de division du travail. Développé en France au XVIII^{ème} siècle, le principe d'interchangeabilité considère que les composants élémentaires pour la fabrication d'un produit fini sont identiques et donc interchangeables [AS93]. On devrait alors pouvoir prendre n'importe quel composant et l'utiliser pour la production de toute instance de produit. Quant à la division de labeur, elle permet une simplification du travail, sa standardisation et sa spécialisation. Ainsi, une activité complexe est subdivisée en tâches élémentaires (opérations) qui peuvent être effectuées séparément sur différents postes de travail.

Ford est sans doute celui qui a révolutionné les systèmes de production en introduisant le principe du flux de produits passant par plusieurs stations. Un exemple qui illustre bien ce cas de figure, est le modèle de voiture T de Ford qui a été produit à plus de 15 millions d'exemplaires entre 1908 et 1927 [Hit96].

La justification théorique de la production de masse a été présentée la première fois en 1910 par K. Bücher. Le principe est simple, en considérant le coût fixe a et le coût variable bx où x est la quantité fabriquée, le coût total est donné par :

[4]PCI Scemm représente une filiale de PCI, elle est centré sur les activités liées à l'usinage et siège à St Etienne. PCI Scemm compte de nombreux clients notamment des constructeurs automobiles et constitue le premier fournisseur de PSA Peugeot Citroën.

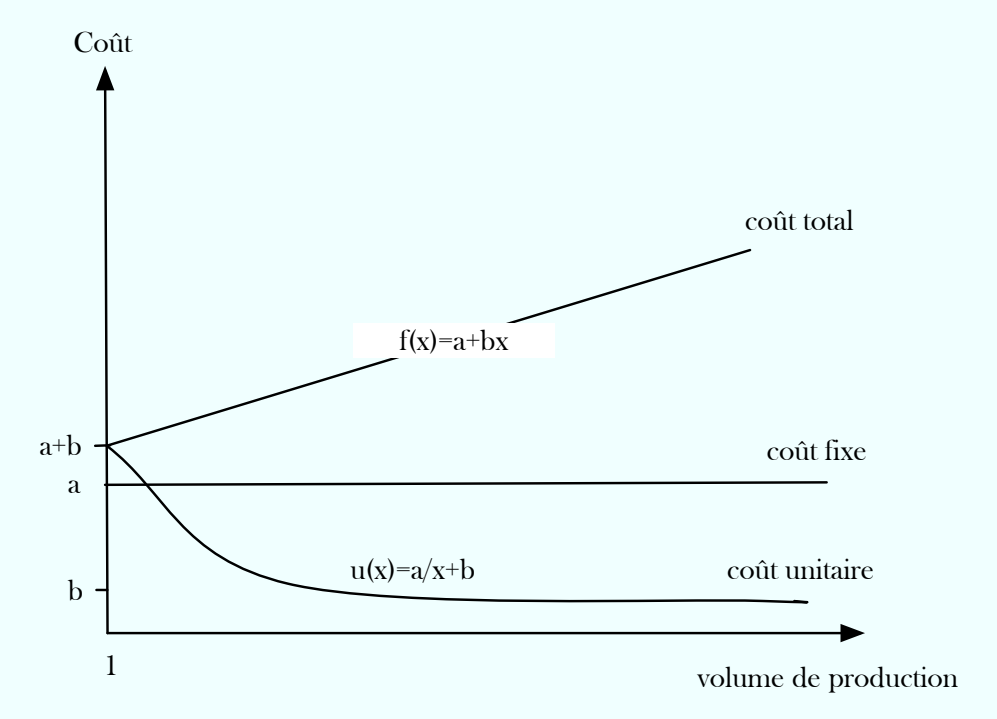


FIG. 1.2 – Principe de la production de masse

$$f(x) = a + bx \tag{1.1}$$

et le coût de production unitaire moyen est donné par :

$$u(x) = a/x + b \tag{1.2}$$

L'équation (1.2) permet de voir clairement que le coût unitaire est directement lié au volume, plus celui-ci est important plus le coût unitaire est faible.

Les lignes d'usinage dédiées sont conçues pour la production d'un seul type de produits ou des variantes proches. Ces lignes sont utilisées pour la grande série, car elles ont une grande capacité de production ce qui engendre un coût de revient unitaire relativement faible.

La mise en place des lignes dédiées présente de nombreux avantages, ainsi nous évoquons les plus importants :

1. Une très grande productivité. En effet, les volumes produits peuvent atteindre des millions de pièces, ceci réduisant le prix de revient [Fin04].
2. Une grande précision dans l'usinage en raison de la fixation des outils.
3. Il y a besoin de moins de pièces de rechange car l'équipement est standardisé.
4. La qualité des produits et la stabilité de leur caractéristiques est sans doute une des spécificités les plus attrayantes de ce type de lignes. En effet, celles-ci ont un haut

degré d'automatisme, ce qui minimise le risque de fabriquer des produits de mauvaise qualité.

5. Ces lignes peuvent fonctionner jusqu'à 30 ans avec des modifications tous les 7 ans environ.
6. L'amortissement de ces lignes se fait dés les premières années, elles ont, de ce fait, un coût de revient très compétitif.

Les lignes dédiées à la production de masse présentent toutefois quelques inconvénients, parmi lesquels :

1. La rigidité du système ne simplifie pas l'évolution de la production vers d'autres types de produits.
2. La mise en œuvre d'un tel système est coûteuse.
3. La capacité de production de ces lignes est fixe et le système n'est pas évolutif, c'est-à-dire sa configuration et son taux de production sont figés pour toute sa durée de vie.
4. Dans la pratique, les lignes dédiées sont sous-utilisées et fonctionnent souvent en sous-régime. Une étude de [MT99] montre que de telles lignes ne sont utilisées en moyenne qu'à 53 % de leur capacité effective. Les auteurs expliquent ce taux d'utilisation par le changement de la demande pour certains produits entre le moment où le produit est conçu et celui où la ligne est mise en fonctionnement. Cette sous-utilisation crée un manque à gagner et monopolise des fonds qui pourraient être fructifiés.

Il est toutefois important de noter que lorsque la demande est importante (atteignant plusieurs millions par an) et stable durant un certain nombre d'années, ce type de lignes devient de loin la meilleure solution en termes de rentabilité. Dans ce cas, l'importance des fonds immobilisés impose une vraie réflexion quant à l'optimisation du coût de l'investissment.

Un des principes des lignes dédiées réside dans la réduction du temps de cycle ce qui a une influence directe sur la réduction des coûts de fabrication unitaires. L'amélioration des flux est également possible en raison de la concentration des opérations uniquement à valeur ajoutée.

Pour conclure, les lignes d'usinage dédiées sont économiques et très rentables à condition que les volumes de production soient importants. Il est également important de souligner que ces lignes se prêtent bien à l'optimisation dans le sens où le critère à optimiser est facile à identifier (minimiser le coût unitaire). Par ailleurs, l'intérêt des lignes dédiées est également lié au fait que leur structure représente une forme d'organisation de base pour les autres systèmes, c'est-à-dire que l'ensemble des problèmes qui se posent lors de l'optimisation d'une ligne dédiée sont également présents lors de la conception des autres types de lignes.

Le champs d'application des lignes dédiées s'est sensiblement réduit depuis les années 80. Il reste néanmoins très important pour nombre de constructeurs de lignes d'usinage, il représente, pour PCI-Scemm par exemple, près de 40% de leur production globale (voir site de PCI, [PCI06]). Par ailleurs, les industriels cherchent des réponses pour faire face aux nouvelles conditions du marché qui est caractérisé par une demande souvent volatile pour

les produits finis. De ces préoccupations sont nés les concepts de *mass customization*, des produits et lignes modulaires composées d'unités standard qui sont facilement remplaçables et interchangeables. Ces systèmes permettent de réduire la diversification en élargissant le champ d'application des lignes dédiées à la grande série.

1.2.3 Les lignes d'usinage flexibles

Les systèmes flexibles (FMS, pour *Flexible Manufacturing Systems*) permettent l'usinage de plusieurs types de produits, appartenant à une famille large. Par famille nous entendons les produits ayant des dimensions proches et des caractéristiques géométriques similaires, ainsi que les mêmes classes de tolérances, afin qu'ils puissent être usinés par le même équipement. Les lignes flexibles mettent en place des machines programmables, la partie logicielle prend en charge les changements éventuels tels que les programmes d'usinage ou l'ordonnancement des produits à fabriquer. L'objectif de ce système est de produire plusieurs types de produits et de faire face à des changements possibles durant le fonctionnement du système en termes de volumes fabriqués. Ces lignes assurent un passage rapide d'un type de produit à un autre mais elles présentent un certain nombre d'inconvénients :

1. Les systèmes flexibles sont très onéreux car ils emploient des machines de type CNC (pour *Computer Numeric Control*). Les machines de ce type sont fabriquées avant la planification des processus. À ce stade, les spécificités de l'usinage ne sont pas encore connues, ce qui amènent les constructeurs à intégrer le plus de fonctionnalités au risque de ne pas les employer par la suite.
2. Le développement de la partie logicielle est également onéreuse.
3. À l'inverse des machines dédiées, qui ne contiennent que des outils fixes mais qui permettent une grande précision, les machines CNC fonctionnent sur le principe de changements fréquents d'outils. La forte flexibilité des machines est directement liée à la possibilité de les reprogrammer à souhait en changeant d'outils au détriment d'une perte de précision, ce qui se répercute sur la qualité des produits et donc sur la productivité.
4. En raison des rapides avancées technologiques, ces machines sophistiquées sont très rapidement sujettes à obsolescence.
5. L'étude menée dans [MUKH02] met en évidence les enjeux économiques pour l'industrie manufacturière qui met en place des systèmes de production flexibles. L'étude met en avant le manque à gagner induit par l'investissement dans les systèmes de type FMS, ce dernier est souvent lourd pour une flexibilité superflue, car deux tiers des industriels déclarent ne pas utiliser leur FMS à son entière capacité. Un questionnaire, fait auprès des industriels, rapporte qu'ils sont 75 % à exprimer leur besoin de passer à un système moins coûteux. Les auteurs suggèrent de répondre à ces besoins avec les systèmes basés sur des machines modulaires avec unités standard et une architecture ouverte afin de permettre une conversion efficace et rapide[5].

[5]La conversion serait possible grâce à l'ajout et à la suppression de modules sans affecter l'ensemble du système.

1.2.4 Les lignes d'usinage reconfigurables

Les systèmes reconfigurables, notés *RMS* (pour *Reconfigurable Manufacturing Systems*), sont introduits dans [KHJ+99]. Les auteurs y justifient le besoin des industries manufacturières, dicté par la conjoncture économique, d'acquérir des systèmes de production capables de s'adapter aux perpétuels changements du marché.

Un système manufacturier reconfigurable est conçu depuis le début pour permettre des changements simples dans sa structure physique afin de répondre à la fluctuation dans la demande en termes de volume et de type de produits [KHJ+99].

Les auteurs précisent que les caractéristiques clés d'un système reconfigurable résident dans les points suivants :

- **Modularité** : les lignes reconfigurables ont une structure physique et logicielle modulaire. Une telle structure simplifie et accélère la reconfiguration. En fait, l'utilisation des machines modulaires composées d'unités standard et des architectures ouvertes facilitent l'ajout et la suppression de modules sans affecter toute la ligne. La modularité est présente à plusieurs niveaux : ainsi, une machine est considérée comme un module à l'échelle de la ligne, de la même façon, une broche est un module pour une tête d'usinage.
- **Intégrabilité** : l'intégration des modules dans le système devient possible au moyen d'interfaces capables de prendre en charge les nouveaux composants.
- **Personnalisation de la flexibilité** : le principe est de fournir la juste flexibilité nécessaire pour la fabrication d'une famille définie et restreinte de produits. Comme les machines sont construites après la définition de la famille des produits il devient possible de fournir la juste flexibilité[6].
- **Ajustabilité** : pour que la ligne puisse être en adéquation avec la demande en terme de capacité de production, l'ajustement des stations permet d'ajouter ou de supprimer des machines, mais également d'augmenter ou de diminuer la capacité d'une machine par la modification de sa structure, en l'occurrence par l'ajout ou la suppression de broches pour une unité d'usinage.
- **Convertibilité** : les fonctionnalités de la ligne doivent pouvoir être transformées pour produire de nouveaux types de produits.
- **Diagnosticabilité** : les défaillances du système doivent pouvoir être détectées automatiquement et corrigées.

Une comparaison des RMS avec différents types de systèmes de production est faite dans [MUK00]. Ainsi, ce paradigme nouvellement introduit devrait pouvoir apporter de nombreux avantages, notamment la flexibilité à juste mesure. Il nous semble pourtant qu'il reste de nombreuses questions auxquelles il faut répondre avant d'atteindre les objectifs visés. Les interrogations qui nous semblent les plus cruciales sont les suivantes :

[6]Contrairement aux lignes flexibles qui offrent une flexibilité absolue mais le plus souvent inutile, le principe des lignes reconfigurable est d'offrir une flexibilité à juste mesure limitée à la fabrication des produits d'une famille restreinte définie au préalable.

- L'absence d'interfaces standards est l'inconvénient majeur. Celles-ci doivent prendre en charge l'architecture des machines modulaires de façon à permettre des évolutions du système.
- Nous n'avons pas suffisamment de recul pour juger de l'efficacité de tels systèmes. En effet, le principe n'a été introduit qu'en 1999, ces systèmes sont donc toujours en phase expérimentale, ils n'ont pas encore pu confirmer ni leur fiabilité ni leur productivité à juste mesure.
- Une imprécision réside dans la définition de la famille de produit. La conception du RMS est définie autour d'une famille restreinte de produits qui nécessite une anticipation sur la demande du marché en termes de type des produits à fabriquer dans l'avenir. À long terme, il semble difficile de faire des prévisions fiables. Nos partenaires industriels et fournisseurs d'équipement pour les constructeurs automobiles nous le confirment : «*il n'a jamais été possible de présager des changements effectifs qui concernent l'évolution des produits*».
- La flexibilité apportée en principe par les RMS est limitée à une famille dont la definition elle-même est imprécise. Cette dernière restreint l'évolution des RMS aux prévisions faites au moment de sa conception qui peut être décalée de la réalité. De surcroît, les produits d'une famille doivent être intrinsèquement proches. Par exemple il faut qu'ils aient en commun un minimum d'opérations [KL98]. Ce dernier point est capital, pourtant il n'a pas été abordé dans les travaux cités.

1.3 La conception des lignes d'usinage

La conception des lignes d'usinage exige une démarche globale qui nécessite la résolution de plusieurs problèmes interconnectés [AS93]. Idéalement, les décisions relatives à tous ces problèmes doivent être considérées simultanément, seulement le problème global est si complexe qu'il devient indispensable de le décomposer en plusieurs phases, chacune répondant à des décisions moins complexes. Les ouvrages de référence en cette matière [AS93, Hit96] proposent plusieurs décompositions possibles. Nous avons opté pour celle offrant le plus de détails [AS93].

La conception du produit doit être réalisée le plus en amont avant la planification des processus et l'étude de l'affectation des opérations de production aux postes de travail. L'étude du flux et des moyens de transport sont également à considérer. Dans la suite, nous décrivons ces étapes plus en détail :

- **La conception du produit** : cette phase correspond à la définition du produit répondant à un besoin du marché. Il s'agit de donner une description formelle et précise d'un produit futur. Des outils de CAO (pour Conception Assisté par Ordinateur repris de *CAD* qui signifie en anglais *Computer Aided Design*) offre un cadre pratique pour réaliser une telle démarche en proposant la possibilité de représenter le produit en trois dimensions avec des points définissant sa géométrie, sa typologie et ses caractéristiques.
- **Planification de processus** (*Process planning*) : cette étape s'intéresse à la définition des gammes de fabrication. Un ordre partiel entre les opérations est notamment établi.

En outre, c'est lors de cette phase que les contraintes technologiques sont définies. Cette étape nécessite une véritable compréhension de la faisabilité des opérations et des besoins fonctionnels décrits lors de la conception du produit.

- **Configuration (ou affectation des opérations)** : À ce stade, on étudie l'affectation des opérations aux stations de travail (agencement logique) de façon à ajuster la productivité du système à la demande. C'est également à ce niveau que l'on détermine les différents outils pour effectuer ces opérations. Lorsque de tels outils peuvent être sélectionnés, il devient impératif de veiller au respect des contraintes définies précédemment entre les opérations, en particulier des contraintes de précédence.
- **Flux et moyen de transport** : afin de permettre une meilleure gestion de flux, il est nécessaire d'étudier le flux des produits dans le système de production et l'agencement physique des équipements (facilities layout) qui concerne la façon de disposer les machines. Quant au choix des moyens de transport, il faut qu'il soit cohérent avec la décision définissant l'architecture du système qui est prise antérieurement. En effet, un convoyeur linéaire conviendra mieux à une structure linéaire du système.

La prise en compte simultanée de l'ensemble de ces problèmes est en pratique très difficile, aussi la démarche usuellement adoptée procède de façon hiérarchique en considérant les étapes les unes après les autres. Il est, cependant, souvent indispensable de revenir sur des étapes antérieures lorsque les décisions qui y sont prises s'avèrent inadéquates. Un exemple de mise en pratique d'une telle démarche est proposée dans [ZZX02], ce schéma est rapporté dans la figure 1.3 qui reprend les étapes décrites précédemment en montrant les différentes interactions possibles.

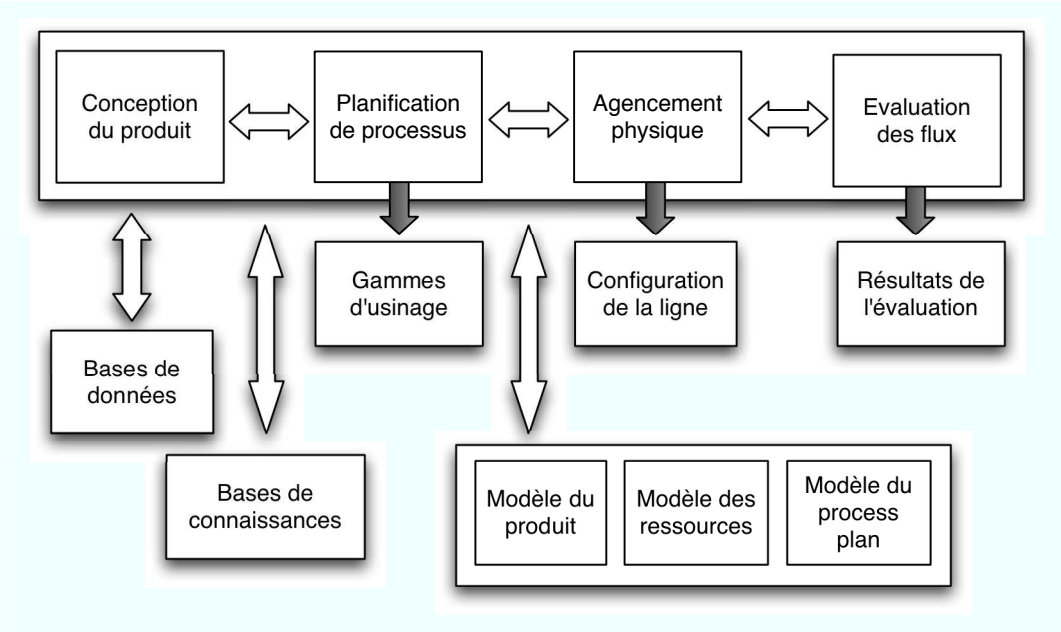


FIG. 1.3 – Démarche globale pour la conception d'une ligne d'usinage

La modélisation des caractéristiques du produit permet de réaliser la description du

produit à fabriquer nécessaire pour les étapes suivantes. Ensuite, la planification de processus permet de proposer plusieurs gammes pour alimenter l'étape de l'agencement physique (qui est représentée par la troisième et quatrième étapes dans notre description ci-dessous). Lors de cette étape une configuration de la ligne est établie en tenant compte de la productivité voulue et du coût des équipements. Enfin, une évaluation des flux et du coût de la ligne est réalisée en prenant compte de l'ensemble des décisions qui ont été prises auparavant. Ainsi, lorsque l'évaluation des résultats n'est pas satisfaisante, le concepteur est en mesure d'apporter des modifications à tous les niveaux allant de la conception du produit jusqu'à l'agencement. De plus, la figure illustre l'échange qui existe lors de la conception du produit avec une base de données dans laquelle sont stockées des solutions type.

La démarche proposée dans cette thèse est similaire à celle de [ZZX02]. Cependant, dans le cas de la figure 1.3, il s'agit d'un atelier flexible pour lequel l'affectation des opérations aux postes de travail est réalisée en temps réel, ce qui explique l'absence de cette phase dans le schéma rapporté. En effet, les centres d'usinage sont en général équipés pour effectuer un maximum d'opérations tandis que pour les lignes dédiées les opérations à affecter sont plus précises dans le sens où il n'y a que les opérations sollicitées pour la fabrication du produit sélectionné qui sont présentes sur la ligne. C'est pourquoi le problème de l'affectation des opérations se pose lors de la conception de la ligne juste après la définition des gammes, c'est-à-dire à l'étape de l'agencement logique et physique définissant la configuration de la ligne.

Nos travaux s'inscrivent dans une démarche globale similaire au schéma précédent. Plus particulièrement, notre intérêt se positionne à cheval entre la phase de planification de processus et celle de l'affectation des opérations aux postes de travail (agencement logique). Ainsi, notre démarche concerne le bureau de méthodes mais aussi celui de la fabrication car nous nous intéressons à la fois au choix d'une gamme mais aussi à l'affectation des opérations aux stations en précisant les équipements à mettre en place.

Jusqu'à présent, des avancées incontestables dans la CAO ont permis de réduire le temps de la conception des produits à l'inverse de la phase de configuration des systèmes de fabrication pour laquelle moins d'efforts ont été fournis. Cette dernière devient le goulot d'étranglement du cycle de vie du produit [Das03]. Il est évident que l'application de méthodes basées sur des modèles mathématiques efficaces réduirait le temps de réalisation de cette phase et permettrait un gain considérable sur le temps de conception global : produit-process-système de fabrication. D'autre part, l'analyse et l'optimisation de la configuration doivent être réalisées le plus en amont possible pour éviter des changements trop tardifs dans la spécification du produit et de son système de fabrication car cela engendrerait des coûts prohibitifs.

C'est dans ce cadre que s'inscrit notre contribution. Plus exactement, nous nous plaçons dans un contexte d'usinage pour apporter des réponses efficaces aux questions relatives à la structuration des lignes dédiées et reconfigurables utilisant des machines multi-broches.

Nous proposons dans la figure 1.4 [DGL+04], un schéma global reprenant la démarche adoptée. La figure permet de situer notre approche dans un contexte plus général en présence des diverses informations qui sont définies dans les autres phases. Ces informations peuvent être déterminées en amont, elles sont alors traitées comme données d'entrée telles

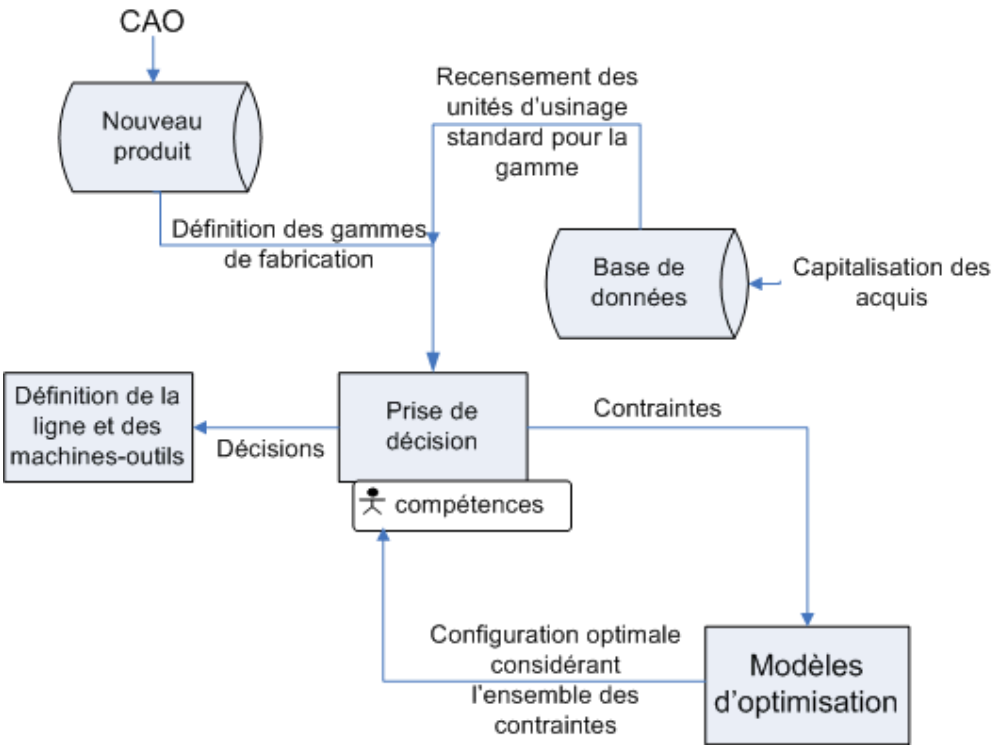


FIG. 1.4 – Schéma global pour l'intégration de méthodes d'aide à la décision

que l'ensemble des unités d'usinage disponibles (qui sont stockées dans une base de données) Ces unités sont caractérisées par leur coût, le temps d'usinage et les opérations effectuées.

Par ailleurs, le concepteur formule des contraintes qui sont inhérentes aux technologies utilisées (gammes d'usinage retenues lors du process plan) mais aussi des contraintes issues de son expérience. Il est intéressant de voir que l'approche proposée peut être intégrée dans un schéma itératif global. Ceci permet en particulier aux décideurs de relancer la résolution pour modifier les structures proposées (solutions fournies) au moyen de l'ajout de contraintes supplémentaires. Ainsi, les décideurs sont en mesure d'intégrer leur savoir-faire et leurs compétences pour mettre à profit leur expérience. L'ensemble des données d'entrée est ainsi constitué pour que le modèle d'optimisation puisse trouver une solution optimale respectant les contraintes fournies. La solution procure en sortie les unités d'usinage qui seront placées sur chaque station et le nombre de stations à mettre en service. Le cœur de cette approche réside dans la modélisation des contraintes et l'optimisation du modèle obtenu, c'est précisément de ces sujets que nous traitons dans cette thèse.

1.4 Enjeux de l'optimisation lors de la configuration des lignes d'usinage

Les enjeux économiques sont importants pour les entreprises manufacturières qui mettent en place des systèmes d'usinage automatisés en raison du coût d'acquisition des unités d'usinage et machines-outils. Ces entreprises doivent investir lourdement dans l'installation et la mise en œuvre des lignes qu'elles fabriquent d'une part et pour sa reconfiguration d'autre part. Ces coûts se répercutent en grande partie sur le prix du produit fini et doivent, par conséquent, être minimisés.

Par ailleurs, dans le cas des lignes dédiées à la grande série dont la structure est figée, une erreur commise à ce stade est quasiment irrémédiable (ou coûterait très cher). À l'inverse, si un gain est réalisé à cette étape, des bénéfices considérables peuvent être obtenus en tenant compte de l'effet d'échelle.

Une fois que les phases de conception du produit à fabriquer et de planification des processus sont en partie réalisées, il faut commencer à déterminer la configuration de la ligne en termes de sélection des unités d'usinage et de leur affectation aux stations.

Qu'il s'agisse de conception préliminaire ou de reconfiguration les problèmes de décision restent les mêmes, à savoir :

1. Quelles sont les unités d'usinage à choisir pour assurer la productivité visée ?
2. Combien de stations de travail faut-il installer afin d'assurer la productivité voulue tout en minimisant le coût de la ligne ?
3. Comment affecter les unités d'usinage aux stations ?
4. Dans quel ordre activer les unités d'usinage sur chaque station ?

Dans ces circonstances, il devient intéressant, voire indispensable, pour les décideurs, de détenir un outil d'aide à la décision leur permettant de trouver la meilleure configuration en termes de coût de mise en place de la ligne comportant les coûts d'acquisition des unités d'usinage et des stations de travail. C'est dans ce cadre que s'inscrivent les travaux rapportés dans cette thèse. La démarche proposée est basée sur une approche exacte employant des modèles mathématiques pour répondre aux problèmes de décision qui se posent lors de la phases du choix de la gamme (process plan) et celle de l'affectation des opérations.

Ces problèmes se posent pour tous les types de lignes d'usinage. En effet, qu'il s'agisse d'une conception préliminaire dans le cas de lignes dédiées ou d'une reconfiguration dans le cas de lignes flexibles ou reconfigurables, il faut choisir un ensemble d'unités pour effectuer la gamme d'opérations sélectionnée. Néanmoins, l'objectif lors d'une conception préliminaire peut être différent de celui à considérer lors d'une reconfiguration. Dans le dernier cas, il peut être pertinent de considérer également le temps et le coût de reconfiguration. Nous attirons l'attention du lecteur sur le fait que le critère de coût considéré, dans notre étude, ne se limite pas au coût de l'investissement uniquement, il peut être de nature diverse. Ainsi, il peut représenter un temps et un coût de reconfiguration ou un tout autre critère.

1.5 Problématique

1.5.1 Descriptif des lignes étudiées

L'objectif de cette thèse est d'apporter les meilleures solutions aux problèmes inhérents à la conception/reconfiguration qui se posent pour les lignes d'usinage. Ces lignes sont dites sérielles (*tandem lines* dans la littérature anglosaxone), et couvrent une large classe des systèmes d'usinage : machine à transfert circulaire, machine à transfert linéaire, ligne de transfert,.... Chaque station est équipée d'une ou de plusieurs unités d'usinage. Les stations sont reliées entres elles à l'aide d'un système de transfert, le plus souvent un convoyeur.

Lorsque la ligne est en fonctionnement, une pièce est positionnée en face de chacune des stations de sorte que les unités d'usinage de ces dernières puissent effectuer leurs opérations. L'activation des unités, qui se matérialise par l'enclenchement de l'ensemble des broches, marque le début d'un cycle et l'arrêt de la dernière tête[7] délimite la fin de ce même cycle. Ainsi, la station goulot (celle qui a le temps d'exécution le plus long) définit le temps de cycle de la ligne. Nous notons L pour désigner la ligne et S_k pour la station avec l'indice k. La figure 1.5 représente le schéma d'une telle ligne.

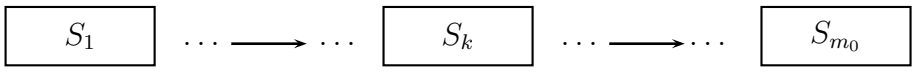


FIG. 1.5 – Schéma d'une ligne

De nombreux travaux existent dans la littérature étudiant les stocks tampons dans ce type de ligne, nous rapportons ici quelques références comme suit : [DG89],[DDX89],[DES06] et [DP06]. Actuellement, les industriels ont tendance à placer les stocks entre les lignes. Ainsi, dans notre étude, nous supposons que les stocks sont uniquement en amont ou/et en aval à la ligne, ce qui nous permet de ne pas en tenir compte lors de l'étude de la configuration de la ligne.

1.5.2 Modes d'activation des unités d'usinage

À présent, nous allons développer plus en détail les unités d'usinages qui composent une station ainsi que leurs différents modes de fonctionnement.

Avant d'aborder les modes d'activation des unités composant une station, il est nécessaire de définir l'unité d'usinage elle-même. Pour ce faire, nous présentons des têtes multi-broches et des têtes mono-broche.

Dans le cas d'une tête multi-broches, plusieurs broches combinées sur une unité d'usinage[8]

[7]Par dernière tête à s'arrêter, nous entendons celle qui a le temps d'exécution le plus long de toute la ligne.

[8]Les termes tête et unité d'usinage peuvent être utilisées indifféremment.

permettent l'exécution de 7 opérations en parallèle. Une telle tête d'usinage est illustrée dans la figure 1.6, elle permet d'exécuter 6 opérations simultanément [9]. L'utilisation d'unités d'usinage multi-broche permet de réduire le temps d'exécution de l'ensemble des opérations. Le temps nécessaire pour effectuer toutes les opérations d'une même tête d'usinage peut être estimé au plus grand temps d'exécution de ses opérations. Toutefois, cette estimation n'est pas la seule et a fait l'objet de nombreux travaux.

FIG. 1.6 – Tête multi-broches de PCI Scemm

Selon le temps de cycle à respecter et la faisabilité technique, il peut être plus judicieux de choisir un type d'activation des unités d'usinage par rapport à un autre. Ce mode régit l'enclenchement des unités d'usinage appartenant à une même station. La décision inhérente à ce choix n'est pas l'objet de l'étude proposée mais elle doit néanmoins être prise en amont à notre étude de manière a pouvoir l'intégrer lors de la modélisation des contraintes relatives au temps de cycle. Les modes d'activation existants sont au nombre de trois, nous les désignons par mode : parallèle, séquentiel et mixte. Plus précisément, nous les décrivons comme suit :

- **Le mode parallèle** : toutes les unités travaillant sur la même station sont activées simultanément. Plus précisément, les unités d'usinage de toutes les stations sont enclenchées en même temps par un mouvement synchrone. Le début d'un cycle correspond au moment où toutes les unités sont enclenchées pour effectuer leurs opérations. Lorsque les têtes de toutes les stations ont terminé l'exécution de leurs opérations, le cycle est achevé et la pièce est transférée à la station suivante où un autre cycle est effectué.
- **Le mode séquentiel** : les unités d'usinage appartenant à la même station sont activées de façon séquentielle. Ainsi, la seconde unité n'est enclenchée qu'une fois que la première unité ait terminé l'exécution de ses opérations. Quand la seconde unité termine, à son tour, son travail la troisième tête est enclenchée et ainsi de suite jusqu'à ce que toutes les têtes d'usinage de la station aient terminé. Enfin, la pièce est transférée à la station suivante, où les opérations de celle-ci sont effectuées pendant un autre temps de cycle.

[9] Lorsqu'il n'y qu'une seule broche sur une tête, celle-ci est dite mono-broche.

La figure 1.7 est un exemple d'une station ayant deux unités d'usinage multi-broche activées de façon séquentielle par rotation du support.

FIG. 1.7 – Une station avec le mode séquentiel (PCI Scemm)

- **Le mode mixte** : la station est composée de plusieurs étages qui s'exécutent en séquence. Chaque étage est défini comme un sous-ensemble d'unités d'usinage activées, elles-mêmes, en parallèle. Ainsi, la pièce arrive sur la ligne et le premier étage de la première station est positionné en face de la pièce, de façon à permettre à ses têtes d'effectuer toutes leurs opérations. Une fois que celles-ci ont terminé, le deuxième étage de la première station est positionné à son tour, face à la pièce qui reste immobile, et les têtes du second étages sont activées. Le processus est réitéré jusqu'à ce que tous les étages de la première station aient terminé. Lorsque tous les étages de toutes les stations ont achevé leurs opérations un cycle est terminé. La pièce est déplacée vers la station suivante et un autre cycle commence.

Le cas mixte est clairement une généralisation des deux premiers cas, il se ramène au cas parallèle lorsque les stations ne comportent qu'un seul étage. Par ailleurs, il revient au cas séquentiel lorsque les étages ne contiennent qu'une seule unité d'usinage.

Nous décrivons à présent l'ensemble des données d'entrée du problème de configuration considéré.

1.5.3 Les données

Les lignes étudiées font intervenir différents types d'opérations telles que : le perçage, le taraudage, le fraisage, l'alésage, etc.

L'ensemble des opérations à effectuer pour l'usinage de la pièce à fabriquer, noté **N**, est déterminé dans les phases antérieures de conception de produit et de process plan (dans un premier temps, nous ne considérons qu'un seul type de produit à fabriquer).

Nous supposons également que nous connaissons **l'ensemble des unités d'usinage** capables d'effectuer une ou plusieurs opérations nécessaires à la fabrication du produit, nous le notons **B**. En fait, selon que l'étude de conception soit préliminaire pour une ligne non existante ou bien qu'elle concerne une reconfiguration, les décideurs sont amenés à constituer différemment l'ensemble des unités d'usinage disponibles. Nous citons les points qui nous semblent les plus fréquents, comme suit :

1. Une étude de disponibilité sur le marché des unités d'usinage. À cette étape, il s'agit de récolter les informations sur les différentes possibilités en termes d'unités d'usinage capables d'effectuer une ou plusieurs opérations nécessaires à la fabrication du produit.
2. En cas de présence de compétences ou d'un savoir-faire en matière de conception et de fabrication d'unités d'usinage, celles-ci peuvent être acquises par ce moyen.
3. Des unités encore opérationnelles peuvent être réutilisées, deux situations se présentent alors, nous les décrivons comme suit :

   (a) Si la ligne est à concevoir, les unités peuvent être récupérées à partir d'autres lignes de production qui ont été mises hors de fonctionnement à condition que les unités soient toujours opérationnelles.

   (b) Si la ligne est à reconfigurer alors les unités avec lesquelles le système fonctionne, en l'occurrence celles qui sont présentes sur la ligne, sont intégrées dans la liste de disponibilité car elles sont toujours valables et candidates pour effectuer les opérations qui leurs correspondent. Néanmoins, si de nouvelles opérations doivent être introduites alors il faut compléter la liste des unités d'usinage disponibles avec des unités capables d'exécuter ces nouvelles opérations. De même, si certaines opérations ne sont plus à effectuer alors la liste des disponibilités est mise à jour et les unités correspondantes sont à écarter.

Dans notre étude, chaque unité d'usinage est décrite par l'ensemble d'opérations qu'elle effectue, son coût et son temps d'exécution (opératoire). Ces paramètres sont pris en compte comme des données du problème.

Nous considérons également des contraintes relatives aux caractéristiques des unités. Celles-ci peuvent représenter la charge à ne pas dépasser, leurs dimensions ou leur capacité de production. Nous résumons, dans ce qui suit, les notations propres aux données des unités d'usinage :

- l'ensemble **B** des unités d'usinage disponibles est supposé connu. Une unité d'usinage est définie par l'ensemble des opérations qu'elle effectue.
- le coût de l'unité $b \in \mathbf{B}$ est donnée par q_b,

- t_b désigne le temps opératoire pour effectuer toutes les opérations de l'unité b.

La sous-section qui suit est dédiée à la contrainte de temps de cycle qui découle directement de la productivité souhaitée. Nous distinguons naturellement les trois modes d'activation en précisant pour chacun d'eux l'expression de temps de cycle en fonction des temps opératoires des unités qui équipent la ligne.

1.5.4 Le temps de cycle

En pratique, la productivité visée est estimée sur la base du volume annuel à atteindre. De ce dernier est déduite la productivité par unité de temps en tenant compte des éventuelles interruptions qui pourraient perturber la production. Une borne supérieure sur le temps de cycle T_0 en est déduite, c'est-à-dire la limite de temps accordé aux stations pour achever toutes leurs opérations. Le temps de cycle effectif (réel) est noté T, il est déterminé par le temps de la station goulot, c'est-à-dire celle qui a le plus grand temps d'exécution. Ainsi :

$$T = \max_{k=1,\dots,m} \{TS_k\}, \tag{1.3}$$

où m est le nombre de stations et TS_k est le temps de travail de la station S_k ; ce dernier dépend du mode d'activation des unités d'usinage et, selon le cas, sa valeur est déterminée ainsi :

1. **Le mode parallèle** : lorsque les unités sont activées de façon simultanée, le temps de travail de toute station est déterminé par l'unité d'usinage qui a le plus grand temps d'exécution parmi toutes celles qui l'équipent, *i.e.*

$$TS_k = \max_{b \in S_k} \{t_b\}, \quad \forall k = 1, \dots, m \tag{1.4}$$

   Le temps de cycle effectif de la ligne correspond au maximum des temps de travail des stations. Dans ce cas précis, T_0 est déterminé par l'unité ayant le plus grand temps d'exécution.

$$T = \max_{k=1,\dots,m} \{\max_{b \in S_k} \{t_b\}\} \tag{1.5}$$

2. **Le mode séquentiel** : les unités sont activées de façon séquentielle au sein de chaque station. Par conséquent, le temps de travail de chacune d'entre elles est égal au cumul des temps d'exécution de ses unités :

$$TS_k = \sum_{b \in S_k} t_b, \quad \forall k = 1, \dots, m \tag{1.6}$$

   On en déduit que :

$$T = \max_{k=1,\dots,m} \{\sum_{b \in S_k} t_b\} \tag{1.7}$$

3. **Le mode mixte** : Nous désignons par s_j^k le j^{me} étage de la k^{me} station. Comme l'exécution de toutes les unités appartenant au même étage se fait en parallèle, le temps opératoire Ts_j^k de l'étage s_j^k est par conséquent déterminé par le plus grand temps d'exécution des unités de cet étage.

$$Ts_j^k = \max_{b \in s_j^k} \{t_b\}, \quad \forall k = 1, \ldots, m \quad \forall j = 1, \ldots, q_0 \tag{1.8}$$

Au sein d'une station, l'exécution séquentielle de ses étages, implique un temps de travail, que l'on note TS_k, qui est égal au cumul des temps de ses étages :

$$TS_k = \sum_{j=1}^{q_0} Ts_j^k, \quad \forall k = 1, \ldots, m \tag{1.9}$$

Tel que q_0 est le nombre maximum d'étages possibles dans une station.
Par conséquent :

$$T = \max_{k=1,\ldots,m} \sum_{j=1}^{q_0} \max_{b \in s_j^k} \{t_b\} \tag{1.10}$$

À présent que nous avons examiné formellement les répercussions engendrées par le choix d'un mode d'activation sur le calcul du temps de cycle, nous décrivons, par la suite, les autres types de contraintes que nous prenons en compte.

1.5.5 Les contraintes

Nous considérons des contraintes qui décrivent des incompatibilités entre les unités d'usinage ne devant pas travailler sur le même poste de travail. Nous introduisons le terme **contraintes d'exclusion** pour y faire référence. Pour illustrer ces contraintes, nous prenons l'exemple d'une station qui a une charge électrique maximale à ne pas dépasser. Pour toutes les combinaisons d'unités d'usinage ayant un cumul de charge supérieure à cette limite, il faut introduire des contraintes interdisant leur affectation simultanée à une station quelconque.

Pour prendre en compte des impératifs de précision nécessitant l'exécution de certaines opérations sur la même station nous introduisons les **contraintes d'inclusion**. Par exemple, si une grande précision est requise pour la distance entre deux trous alors il devient impératif d'effectuer les deux opérations de perçage sur la même station. En particulier, si chaque trou est effectué sur une station distincte, les chances de parvenir à la précision requise seront fortement compromises en raison du transfert de la pièce entre les stations.

Par ailleurs, certaines opérations requièrent une exécution postérieure à l'accomplissement d'autres opérations. Ces contraintes définissent un ordre partiel entre l'ensemble des opérations. Il s'agit alors des **contraintes de précédence**. Par exemple, avant d'effectuer une opération de perçage, une encoche est, le plus souvent, requise afin de procéder au perçage sans dommage.

Une limite sur **la capacité de la ligne** en nombre de stations est également prise en compte. Celle-ci représente une limite sur l'espace disponible pour la mise en place de la ligne, comme elle peut représenter une charge totale à ne pas dépasser.

De même, **la capacité des stations** en termes d'unités supportables est également prise en considération. Ainsi, le nombre maximum d'emplacements, prévus par le constructeur, pour équiper les stations, peut également être intégré dans le modèle au moyen de ces contraintes.

1.6 Conclusion

Dans ce chapitre, nous avons abordé les systèmes de production de façon générale, puis nous avons mis l'accent sur les lignes d'usinage de façon plus particulière car ce sont ces dernières qui nous intéressent. Nous avons introduit la problématique qui fait l'objet de notre étude et qui se pose pour les trois types de lignes abordées, à savoir : les lignes d'usinage dédiées, les lignes d'usinage flexibles et les lignes reconfigurables. Nous posons le problème comme la sélection d'un ensemble d'unités d'usinage pour fabriquer un produit donné en prenant en compte plusieurs types de contraintes et minimisant le coût total. Le présent problème, tel qu'il a été posé, n'a fait l'objet d'aucune étude au préalable, sa modélisation et sa résolution sont pourtant capitales au vu de l'importance des enjeux économiques. Dans cette thèse, nous proposons d'y apporter quelques réponses en abordant le problème avec des approches exactes en raison du gain potentiel qui peut être réalisé avec les solutions optimales.

À présent, nous allons introduire quelques méthodes d'optimisation afin de pouvoir présenter par la suite les travaux de la littérature que nous avons jugés proches de la problématique introduite dans le présent chapitre.

Chapitre 2

Méthodes pour l'optimisation combinatoire

Dans le précédent chapitre nous avons présenté un problème inhérent à la conception /reconfiguration des lignes d'usinage sans évoquer les moyens offerts par la recherche opérationnelle pour le résoudre. Dans le présent chapitre, nous présentons d'abord une introduction à la théorie de la complexité pour sensibiliser le lecteur à la difficulté du problème posé. Ensuite, nous rapportons les méthodes les plus utilisées pour ce type de problèmes. Nous développons particulièrement les schémas basés sur les procédures de type séparation et évaluation car ce sont celles-ci que nous employons dans nos travaux.

2.1 Théorie de la complexité

La théorie de la complexité s'intéresse de façon générale aux problèmes de l'optimisation combinatoire et de façon plus particulière aux problèmes de décision qui leur correspondent. Les problèmes d'optimisation se caractérisent par des contraintes à satisfaire et un objectif à minimiser ou à maximiser. Pour les résoudre à l'optimum, il est toujours possible de parcourir tout l'espace de recherche en analysant toutes les solutions réalisables[1] pour ne garder que la meilleure du point de vue de l'objectif défini.

Un problème de décision est un problème pour lequel il n'y a que deux réponses possibles : oui ou non. Le problème de décision compare, en général, une valeur fixe à la valeur que peut prendre la fonction objectif. La procédure la plus basique pour y répondre est de parcourir tout l'espace des solutions réalisables (ce qui correspond au pire des cas). Ce processus permet, entre autres, de trouver une (voire plusieurs) solution(s) satisfaisant à la condition, dans ce cas, le processus de recherche est arrêté et la réponse donnée est positive. Toutefois si aucune solution parcourue n'est faisable, après que la totalité de l'espace de recherche ait été parcouru, alors une telle valeur n'existe pas et la réponse à donner est négative.

[1]Pour les problèmes difficiles sur lesquels nous reviendrons par la suite, cet espace est large et le nombre de solutions réalisables est très grand, c'est pourquoi il faut trouver le moyen d'éviter de parcourir la totalité des solutions.

Au sens de l'effort de calcul nécessaire, les problèmes d'optimisation sont au moins aussi difficile que les problèmes de décision qui leur correspondent. En effet, la résolution à l'optimum des problèmes d'optimisation nécessitent une énumération complète de l'espace de recherche (à moins d'avoir une preuve de la sous-optimalité de certains espaces qui pourraient, de ce fait, être ignorés). Par contre, pour des problèmes de décision, celle-ci dépend essentiellement de la valeur de l'objectif recherchée, ainsi l'exploration s'arrête dès qu'une telle solution est trouvée. Par conséquent, au mieux le problème de décision est plus facile que le problème d'optimisation correspondant. Au pire des cas, il lui est équivalent. Ainsi, même si la théorie de la complexité a été conçue pour les problèmes de décision, ses résultats sont aussi valables pour les problèmes d'optimisation correspondants.

2.1.1 Complexité d'un algorithme

La complexité d'un algorithme est définie par le temps de calcul nécessaire à son exécution pour la résolution d'un problème d'une taille donnée. Ce temps est, en réalité, exprimé en fonction du nombre d'instructions élémentaires nécessaires à l'exécution de l'algorithme par le processeur. Le choix de cette mesure est justifié par le fait que le nombre d'instructions est indépendant de la puissance des machines qui détermine le temps de calcul effectif. La théorie de la complexité s'intéresse, entre autres, à l'ordre de grandeur de la complexité dans le pire des cas qui est dite en grand $\mathcal{O}$. Pour estimer la complexité d'un algorithme, en grand $\mathcal{O}$, il suffit de borner le nombre d'instructions élémentaires qu'il engendre. Cette borne est exprimée en fonction de la taille des données n, et lorsqu'elle s'écrit en $p(n)$ tel que $p(n)$ est une fonction polynomiale, l'algorithme est dit polynomial et sa complexité est donnée par $\mathcal{O}(p(n))$. Notons que lorsque celle-ci est linéaire la complexité est également dite linéaire. Il existe d'autres types de complexité, par exemple : l'exponentielle soit en $\mathcal{O}(a^n)$. Les algorithmes de complexité en $\mathcal{O}(n!)$ ou en $\mathcal{O}(n^{logn})$ sont également considérés comme des algorithmes de complexité exponentielle.

Jusqu'à présent, nous n'avons abordé que la complexité des algorithmes de résolution, cependant, il est important de la distinguer de la complexité des problèmes eux-mêmes, et ce même si la complexité d'un problème est directement liée à celle de ses algorithmes. De manière informelle, la complexité d'un problème est déterminée par celle du meilleur algorithme connu qui le résout [Pas05]. Dans la suite, nous expliquons plus en détail ce lien en distinguant les deux classes de problèmes qui en découlent, la classe P qui contient les problèmes faciles et la classe NP qui comportent ceux qui sont dits difficiles.

2.1.2 La classe P

Cette classe comporte les problèmes pour lesquels il existe un algorithme de résolution de complexité polynomial, *i.e.* en $\mathcal{O}(p(n))$ tel que $p(n)$ est une fonction polynomiale. Les problèmes de cette classe sont dits faciles et leurs algorithmes de résolution sont souvent efficaces. Toutefois, en pratique, si la fonction polynomiale fait intervenir de haut coefficients avec un degré fort alors un algorithme exponentiel peut être plus efficace. Par exemple, un

algorithme en $\mathcal{O}(2^n)$ peut être plus efficace qu'un autre en $\mathcal{O}(n^{100})$. Si on prend l'exemple de $n = 10$ alors le premier nécessite 1024 instructions tandis que l'algorithme polynomial en engendrera 10^{100}.

2.1.3 La classe NP

Contrairement à ce qu'on peut imaginer, NP ne signifie pas Non Polynomial mais Polynomial Non déterministe. Les problèmes appartenant à cette classe sont définis comme des problèmes qui peuvent être résolus avec des algorithmes non déterministes.

Dans [GJ79], les auteurs proposent une définition pour les problèmes de décision appartenant à cette classe. Ils supposent connaître à l'avance une solution du problème, sans vraiment se soucier de la façon dont elle a été fournie. Dans ce cas, ils suggèrent de vérifier la faisabilité de la solution providentielle et si cette vérification se fait en un temps polynomial alors le problème correspondant appartient à la classe NP.

Les résultats de la théorie de la complétude sont basés sur la conjecture : P$\neq$NP et s'intéresse aux problèmes de décision appartenant à NP-P dits NP-complet. Toutefois, et d'après les définitions des deux classes, nous avons a priori : P$\subseteq$ NP, car toute instance qui peut être résolue avec un algorithme déterministe peut l'être avec un algorithme non déterministe.

Les problèmes d'optimisation qui nous intéressent sont des problèmes dit NP-difficiles, c'est-à-dire ceux pour lesquels les problèmes de décision correspondant sont dit NP-complets. Pour montrer qu'un problème p_1 est NP-complet, il suffit de trouver un autre problème NP-complet p_2 tel que le problème p_2 se réduit polynomialement à p_1. La réduction polynomiale nécessite de pouvoir transformer les données d'une instance de p_2 en des données d'une instance de p_1 avec un nombre polynomial d'étapes. De plus, il faut assurer qu'une solution de p_1 peut être également obtenu, après une transformation polynomiale d'une solution de p_2 (nous référons à l'ouvrage de [GJ79] pour plus de détails sur ce sujet).

2.1.4 Retombées pratiques

Lors de la résolution des problèmes NP-difficiles dont la complexité est exponentielle[2] nous sommes assez vite confrontés à une explosion combinatoire lorsque la taille des instances croît. Afin de sensibiliser le lecteur à la notion de l'explosion combinatoire, nous avons choisi d'apporter un exemple qui nous paraît des plus percutants et qui illustre bien ce phénomène. Soit le nombre qui est égal à : $2^{2^{2^{2^2}}}$, la valeur de ce nombre est supérieure à l'estimation du nombre d'atomes dans l'univers[3] [Col02]. Ainsi, il suffit de combiner cinq fois des deux en exposant pour atteindre des nombres astronomiques au sens propre du terme.

[2]Rappelons que cette classe comportent des algorithmes ayant une complexité en factoriel ou encore en $\mathcal{O}(n^{logn})$.

[3]Pour calculer ce nombre, il faut procéder de haut en bas, *i.e.* décomposer le nombre à partir de 2^2 du sommet ce qui fait 4, puis effectuer 2^4 ce qui fait 16, puis $2^{16} = 65536$, et enfin 2^{65536}.

Toutefois, la complexité d'un algorithme est mesurée dans le pire des cas, il est donc possible de résoudre certaines instances en un temps polynomial. Il y a des algorithmes qui sont efficaces dans la pratique, c'est le cas de l'algorithme du *simplex* qui a une complexité exponentielle [KM72] mais qui permet en pratique de résoudre en un temps polynomial même des instances difficiles.

Le choix d'une méthode pour la résolution d'un problème difficile doit se faire en fonction des objectifs à atteindre et doit prendre en compte la structure du problème. En effet, certaines approches se prêtent mieux à la résolution d'un type de problèmes par rapport à d'autres parce qu'elles conviennent mieux à leur structure. Les approches pour la résolution des problèmes de la classe NP peuvent être classées en deux catégories : les méthodes exactes et les méthodes approchées. Une méthode qui apporte une ou plusieurs solutions optimales en fournissant la preuve de l'optimalité est une méthode exacte. Lorsque l'enjeu de trouver la meilleure solution est important, que la taille des instances n'est pas très grande et que le temps alloué au calcul n'est pas restrictif, l'emploi d'une méthode exacte est envisageable.

Dans le cadre de cette thèse, nous nous sommes focalisés sur les méthodes exactes, c'est pourquoi nous les décrivons plus en détails dans la suite. Dans le paragraphe suivant, nous abordons une méthode de résolution lorsque les variables sont de type réel. Nous reviendrons par la suite sur les méthodes de résolution employant des variables entières et/ou binaires.

2.2 Programmation linéaire

Un programme linéaire permet de définir l'objectif à minimiser ou à maximiser[4] comme une expression linéaire des variables de décision (voir 2.1). Quant aux contraintes elles sont exprimées sous forme d'inégalités (voir 2.2) qui sont des expressions linéaires bornées[5]. Un programme linéaire (PL) peut être exprimé comme suit :

$$min \quad cx \tag{2.1}$$

$$s.c. \quad Ax \geq b \tag{2.2}$$

$$x \in \mathbb{R}^n_+ \tag{2.3}$$

avec $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{mn}$

2.2.1 Résolution des programmes linéaires

L'algèbre linéaire offre une possibilité de résolution des programmes linéaires en se basant sur leur interprétation géométrique. En fait, l'espace des solutions réalisables S du programme linéaire correspond à un polyèdre de l'espace euclidien $\mathbb{R}$ lorsque celui-ci est non vide et non borné (si l'espace est non vide et borné c'est d'un polytope dont il s'agit, voir figure 2.1) [NW98]. L'espace S, appelé plus communément enveloppe convexe, est défini en

[4]max cx est équivalent à min (-cx)

[5]Notons que lorsque les deux bornes sont égales, les inégalités se réduisent à des égalités (équations).

fonction de ces facettes qui correspondent aux contraintes du PL et de ses points extrêmes qui sont définies par l'intersection des contraintes, les contraintes délimitant un point sont dites saturées à ce point, c'est le cas de la facette1 aux points p_1 et p_5 de la figure 2.1.

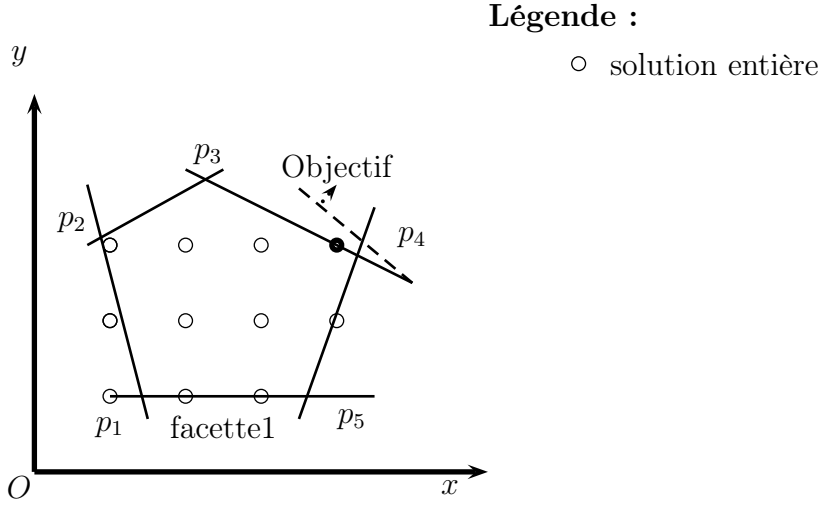


FIG. 2.1 – Enveloppe convexe

Lorsque S est non vide, le programme linéaire admet une ou plusieurs solutions réalisables, la meilleure solution coincide forcément avec un des points extrêmes (sommets) de l'enveloppe convexe de S, car quelle que soit la direction induite par l'objectif la meilleure valeur sera toujours sur la frontière de l'enveloppe convexe. Il arrive toutefois qu'il y ait plusieurs solutions optimales, dans ce cas, ces solutions correspondent à tous les points d'une facette.

Une façon de retrouver la meilleure solution est de parcourir les points extrêmes pour n'en garder que le meilleur du point de vue de la valeur de cx. C'est le raisonnement qui est proposé par l'algorithme du simplex qui a été introduit par [Dan51] dans sa forme primale. L'algorithme du simplex permet d'enclencher un processus itératif pour passer d'un point extrême à un autre en améliorant la qualité des solutions au fur et à mesure de l'avancement. La complexité du simplex n'est pas polynomial, néanmoins il est très efficace en pratique.

Pour certaines structures de problèmes, il est plus intéressant de faire coopérer les deux approches car le dual peut s'avérer plus facile à résoudre.

2.2.2 Programmes linéaires en nombres entiers

Lorsque les variables sont toutes entières le programme linéaire est dit en nombres entiers, il est noté PLNE. Un tel programme peut être exprimé par (2.4)-(2.5).

$$min \quad cx \tag{2.4}$$
$$s.c. \quad Ax \geq b \tag{2.5}$$
$$x \in \mathbb{N}_+^n \tag{2.6}$$

Pour ne considérer que les rationnels, on suppose : $c \in \mathbb{Z}^n, b \in \mathbb{Z}^m, A \in \mathbb{Z}^{mn}$

De façon plus générale, et selon la nature du problème, les variables de décision peuvent être de différents types, on dira que le programme est en variables mixtes s'il y a, à la fois, des variables réelles et entières. On note PLVM pour *Programme linéaire en variables mixtes* [NW98].

$$min \quad cx + hy \tag{2.7}$$
$$s.c. \quad Ax + Gy \geq b \tag{2.8}$$
$$x \in \mathbb{Z}_+^n, y \in \mathbb{R}_+^p \tag{2.9}$$

tel que : $c \in \mathbb{Z}^n, h \in \mathbb{Z}^p, b \in \mathbb{Z}^m, A \in \mathbb{Z}^{mn}, G \in \mathbb{R}^{mp}$

À l'inverse du PL, la solution optimale du PLNE n'est pas forcément contenue sur la frontière de son enveloppe convexe. En raison de l'intégralité des variables, les solutions réalisables peuvent ne pas se trouver sur la frontière définie par S. Ainsi, l'espace de solutions du PLNE est contenu dans S (l'enveloppe convexe du PL correspondant, voir la figure 2.1). Lorsque les contraintes d'intégralité sont relâchées le problème en nombres entiers se ramène au problème (PL) correspondant, il s'agit alors de relaxation du PLNE en PL et la valeur optimale du PL constitue une borne inférieure pour le PLNE correspondant.

Il existe plusieurs procédés pour trouver une solution optimale à un PLNE, parmi ceux-ci, nous citons les procédures arborescentes qui permettent le parcours total de l'espace de recherche par énumération. Ce schéma global gagne en intérêt lorsqu'il est amélioré en employant des techniques de détection de régions sub-optimales afin de les écarter de la recherche.

La procédure par séparation et d'évaluation (PSE), connue sous l'appellation anglo-saxonne de *Branch and Bound*, en est un exemple. Notre intérêt se porte particulièrement sur cette méthode car son caractère générique permet de l'employer dans plusieurs cadres sur lesquels nous reviendrons plus en détails dans les sections qui suivent (elle est notamment à la base des solvers des outils d'ILOG que nous utilisons dans nos travaux). Nous décrivons le schéma global d'une telle approche en mettant l'accent sur les techniques les plus utilisées pour améliorer les performances de cette approche en réduisant l'espace de recherche.

2.3 Procédure par séparation et évaluation

C'est une méthode générique d'énumération implicite qui exploite la structure spécifique des problèmes à résoudre pour réduire l'effort de recherche. L'énumération se fait en dé-

composant le problème original en sous-problèmes chacun séparant l'espace de recherche en plusieurs sous-espaces. Plus exactement, c'est lors d'un branchement qu'une décision est prise déterminant une décomposition en plusieurs sous-problèmes. Les branchements se font au fur et à mesure de la construction de l'arbre en commençant de la racine où aucune décision n'a été prise jusqu'aux feuilles potentielles. Par feuilles nous désignons les nœuds qui n'ont pas de fils possibles (lorsque toutes les décisions ont été prises) et par les feuilles potentielles, nous entendons celles qui peuvent aboutir à des solutions réalisables car toutes les décisions qui sont prises ne mènent pas forcément à des solutions réalisables. Ainsi, des contradictions avec certaines contraintes peuvent être générer, dans ce cas, le processus de recherche est abandonné en coupant la branche correspondante et en reprenant le développement des autres branches. Pour plus de détails sur les PSE nous référons aux ouvrages suivants : [Min83], [Sak84], [DMM97] et [NW98].

Les performances d'un arbre de séparation et d'évaluation sont directement liées à sa taille ; en général, plus sa taille est réduite plus l'approche est efficace. Il faut toutefois noter que l'approche peut être efficace pour des arbres relativement grands lorsque les traitements effectués aux nœuds ne nécessite pas de longs temps de calcul. Cependant, pour réduire la taille de l'arbre, des techniques d'évaluation de solutions permettent de déduire que certaines branches sont non prometteuses. Un mécanisme d'élagage peut ensuite être appliqué afin de «couper» les branches infructueuses qui ont été détectées. Pour ce faire, il suffit de détenir une solution réalisable et une borne de la meilleure valeur que pourrait prendre la fonction objectif au vu des décisions déjà prises, et ce quelles que soient les décisions à venir. La borne utilisée est une borne inférieure dans le cas d'un objectif à minimiser et une borne supérieure dans le cas contraire. Dans la suite, ne nous considérerons que le cas où l'objectif est à minimiser.

Le calcul de la borne inférieure est effectué à chaque nœud de l'arbre, il s'agit alors de la valeur du meilleur coût, que pourrait avoir toute solution développée à partir de ce nœud. Lorsque la valeur de la borne inférieure atteint ou dépasse la valeur de la meilleure solution réalisable rencontrée auparavant, alors aucune solution obtenue à partir de ce nœud ne peut correspondre à la solution optimale. Le mécanisme de l'élagage peut alors opérer en coupant cette branche infructueuse permettant ainsi de réduire l'effort de recherche.

Plus les techniques de calcul de la borne exploitent la structure spécifique du problème traité, meilleure est la qualité de la borne inférieure. Néanmoins, l'obtention de la borne ne doit pas susciter un temps de calcul trop élevé car le cumul de ces temps sur l'ensemble de l'arbre peut ralentir sa construction, ce qui aurait pour conséquences de réduire de façon significative les performances globales du schéma de résolution. Il est donc important de trouver la juste mesure entre l'effort à fournir dans le calcul de la borne et sa qualité.

2.4 Programmation dynamique

L' approche de programmation dynamique est basée sur les principes d'optimalité introduits dans [Bel57, BD62]. En pratique, il faut trouver une expression récursive décomposant le problème en plusieurs sous-problèmes [DM67]. Le principe est de résoudre le problème

initial à partir des solutions des sous-problèmes, pour plus de détails se référer à [NW98]. Ce processus commence d'un état initial x_0 qui permet de passer à un état suivant x_1 en prenant une décision u_1, ce processus est réitéré jusqu'à l'obtention d'un état terminal x_n après avoir pris la décision u_n. Ainsi, l'ensemble de décisions est noté $(u_1, \ldots, u_n)$, celui-ci engendre un coût $f(u_1, \ldots, u_n)$, tel que :

$$f(u_1, \ldots, u_n) = \sum_{i=1}^{n} f_i(x_{i-1}, u_i) \tag{2.10}$$

L'équation (2.10) montre que la prise d'une décision u_i correspondant à un état x_{i-1} engendre un coût $f_i(x_{i-1}, u_i)$ qui ne dépend ni des décisions antérieures ni des décisions ultérieures excepté le fait d'amener à l'état x_i.

2.5 Programmation par contraintes

2.5.1 Définition

Pour expliquer le principe de la programmation par contraintes, nous utilisons le *Problème de Satisfaction de Contraintes* (CSP). Une instance de ce problème est définie par un ensemble de variables $X = \{x_1, \ldots, x_n\}$. À chaque variable x_i correspond un domaine d_i contenant toutes les valeurs que peut prendre cette variable, ainsi l'ensemble des domaines est représenté par $D = \{d_1, \ldots, d_n\}$. Il existe plusieurs contraintes qui lient les variables entre elles $C = \{c_1, \ldots, c_m\}$ telles qu'une contrainte c_j combine plusieurs variables, on écrit : $c_j = \{x_{j_1}, \ldots, x_{j_n}\}$ et que la combinaison des valeurs de ses variables définit le domaine de la contrainte soit $dc_j = d_{j_1} \times d_{j_2} \times \ldots \times d_{j_n}$. Une variable x_i est dite *instanciée* lorsqu'une valeur appartenant à son domaine lui est assignée. L'instanciation est dite *complète* si toutes les variables sont instanciées, elle est *partielle* autrement. De plus, l'instanciation complète est *une solution réalisable* si les valeurs d'instanciation font que toutes les contraintes soient respectées. De même, l'instanciation partielle est une solution partielle si toutefois toutes les contraintes faisant intervenir des variables instanciées sont respectées.

2.5.2 Propagation des contraintes

À la différence de la plupart des approches pour la résolution des problème d'optimisation, la programmation par contraintes ne limite pas l'utilisation des contraintes à la validation des solutions, celle-ci les exploitent et les fait participer activement dans le processus de résolution [BLPN03], cette technique est appelée *propagation des contraintes*. Elle consiste en un mécanisme de déductions qui tend à inférer des contraintes afin de réduire les domaines des variables, on parlera alors de *filtrage des domaines*.

Prenons un exemple d'un CSP binaire qui fait intervenir deux variables x_1 et x_2 telles que leur domaine est égal respectivement à $d_1 = \{0, 1, 2\}$ et $d_2 = \{0, 1, 2\}$:

Si on considère la contrainte suivante : $x_1 < x_2$

La propagation de cette contrainte est obtenue en déduisant les valeurs de chacune des variables qui ne peuvent participer à une solution quelque soit la valeur de l'autre variable. En l'occurrence toute valeur de d_1 qui est supérieure ou égale au maximum de x_2 violera la contrainte, elle est donc inconsistante et doit être supprimée de d_1. Le même raisonnement au regard de la seconde variable x_2 permet de déduire que la valeur 0 ne peut en aucun cas satisfaire à la contrainte et est donc à soustraire de d_2. Intuitivement nous avons appliqué pour le filtrage effectué les règles suivantes :

$$d_1 \longleftarrow d_1 - \{\forall x_i \in d_1 | x_i \geq max(d_2)\} \Longleftrightarrow d_1 \longleftarrow d_1 - \{2\} = \{0, 1\}$$
$$d_2 \longleftarrow d_2 - \{\forall x_i \in d_2 | x_i \leq min(d_1)\} \Longleftrightarrow d_2 \longleftarrow d_2 - \{0\} = \{1, 2\}$$

Ces règles permettent d'assurer la consistance des domaines d_1 et d_2.

2.5.3 Démarche globale

La propagation des contraintes pour un problème NP-difficile est, en pratique, une méthode de résolution incomplète, dans le sens où il subsiste souvent plusieurs possibilitées d'instanciation pour les variables, ce qui fait que la propagation à elle seule ne suffit pas pour détecter toutes les inconsistances de l'espace de recherche. Pour compléter ce schéma, il est donc nécessaire de faire participer une technique de recherche pour instancier les variables aux valeurs résiduelles. Pour ce faire, la méthode usuelle est un algorithme de recherche arborescente pour lequel il faut :

1. définir la politique de branchement, en l'occurrence les contraintes à rajouter en termes d'instanciation de variables ou encore de domaines à subdiviser ;
2. déterminer une politique de retour en arrière *backtracking*, c'est-à-dire expliciter les décisions à prendre lorsqu'une incohérence est détectée. Plus exactement, quelle variable et quelle alternative d'instanciation doit être considérée ?

Les travaux de [SS77] ont introduit la séparation entre les méthodes de déduction (dans le cas de la programmation par contraintes la méthode de déduction est la propagation de contraintes) et les algorithmes d'exploration de l'espace de recherche. La représentation des contraintes, en accord avec la règle de programmation logique, introduite par Kowalski [Kow79], stipulant : algorithme = logique + contrôle.

L'ensemble est repris dans le schéma global, proposé dans [BLPN03], d'une approche basée sur la programmation par contraintes séparant la procédure de recherche et la propagation de contraintes que nous rapportons dans la figure 2.2. Ce schéma montre la séparation entre la définition du problème qui fournit les contraintes initiales et le processus de résolution qui fait coopérer la propagation de contraintes et l'algorithme de résolution représenté par une heuristique de recherche (backtracking). Le schéma met en avant l'ajout de contraintes additionnelles suite à des prises de décision dans l'algorithme de recherche d'une part et à

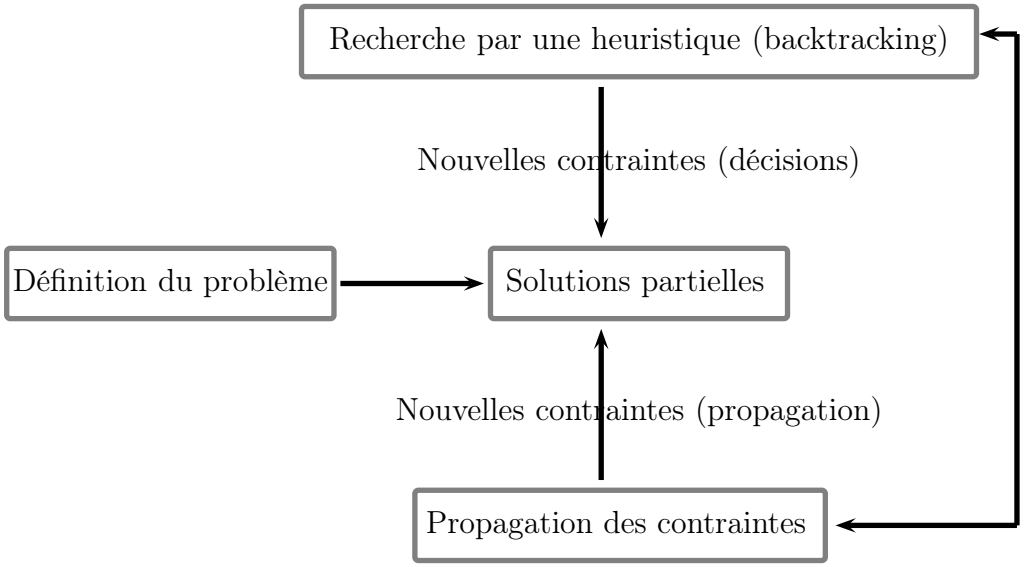


FIG. 2.2 – Interaction entre les différents intervenants d'une résolution par PPC

la propagation de contraintes inférant ces nouvelles décisions d'autre part (c'est l'arc qui lie la recherche par une heuristique et la propagation des contraintes).

Dans la suite, nous décrivons brièvement les méthodes approchées les plus importantes pour souligner notamment les différences avec les méthodes exactes vues jusque-là.

2.6 Méthodes approchées

L'objectif des méthodes approchées est de fournir des solutions de bonne qualité, voire optimales sans se soucier d'apporter la preuve de l'optimalité des solutions. Les méthodes approchées mettent en œuvre des règles heuristiques qui sont spécifiques aux problèmes traités.

Les règles heuristiques peuvent être intégrées dans des approches plus globales, dites méta-heuristiques. Contrairement aux règles heuristiques, les méta-heuristiques ont un caractère générique indépendant des spécificités des problèmes traités. [OL96] les définit comme étant un processus itératif de génération dirigeant une heuristique subordonnée et combinant plusieurs techniques afin d'exploiter et explorer l'espace de recherche en mettant en œuvre des stratégies pour structurer l'information dans le but d'atteindre des solutions proches de l'optimum.

Les méta-heuristiques peuvent être classées en méthodes amélioratives ou constructives. Les algorithmes génétiques sont un exemple du premier type ; cette méthode procède à partir d'une population initiale qu'elle améliore du point de vue de la valeur de la fonction objectif. Les mécanismes pour explorer et exploiter l'espace de recherche sont mis en œuvre grâce aux opérateurs de croisement et de mutation. Nous référons aux ouvrages suivants pour plus de

détails à ce sujet [Gol89, Dav91, Fal96, Koz92, Mul93].

La recherche tabou est également considérée comme une des méta-heuristiques les plus performantes. Cette méthode est un processus itératif explorant l'espace de recherche en passant d'une solution à une autre. La méthode tabou emploi un mécanisme nommé liste tabou afin d'éviter de rester bloquer dans des optimums locaux. L'idée est de garder en mémoire un certain nombre de solutions parmi les dernières parcourues dans le but de ne pas y revenir [DP06]. Il est toutefois permis de garder une solution de moindre qualité car celle-ci peut mener, moyennant une exploration de voisinage, à une bonne solution [GL97].

Par ailleurs, les colonies de fourmis, qui constituent une méthode constructive, permettent de trouver des solutions en s'inspirant du principe de base des fourmis. Le mécanisme met en place un système de collaboration de plusieurs agents (fourmis) pour permettre aux meilleurs de contribuer aux solutions futures (qui seront construites lors des itérations qui suivent) via une mémoire commune : la phéromone, grâce à laquelle une exploitation de l'espace de recherche est accomplie [Bru93, Tai95, DG97]. L'exploration est ajustée au moyen de paramètres qui règlent la proportion du choix aléatoire par rapport aux acquis par expérience. Il est à signaler que cette méthode est récente, il est donc difficile de juger de son efficacité contrairement aux deux méthodes précédentes.

Il est toutefois intéressant de noter qu'une méthode exacte tronquée est une méthode approchée. Ainsi, lorsqu'un algorithme exact est interrompu avant la fin de son déroulement, c'est-à-dire qu'il n'a pas apporté la preuve de l'optimalité de la solution trouvée celle-ci devient de ce fait une solution approchée.

2.7 Conclusion

Nous avons présenté dans ce chapitre un panorama des méthodes exactes et approchées qui sont utilisées dans l'optimisation combinatoire.

Chaque type de méthodes présente des inconvénients et des avantages qui lui sont propres selon les objectifs visés il peut être plus judicieux d'employer l'une ou l'autre. Concernant les méthodes approchées leur principal point faible réside dans le manque de stabilité des algorithmes au vu de la qualité des solutions apportées. Le plus souvent, ces méthodes ne garantissent aucun écart maximum de l'optimum risquant même de ne jamais converger. Même lorsqu'une méthode approchée est efficace pour un type de problème, il reste difficile de garantir que la qualité des résultats obtenus soit la même pour des données différentes. Dans cette thèse, nous avons fait le choix d'une résolution exacte car quand elle devient possible, son emploi est préférable aux méthodes approchées. De plus, pour les lignes dédiées, nous pouvons accorder du temps à la résolution du problème traité car c'est une décision qui n'intervient qu'une fois durant la vie du système lors de la conception préliminaire et une fois tous les 7 ans environ lors d'une reconfiguration.

À présent que nous avons introduit les méthodes de résolution de façon générale, nous consacrons le chapitre suivant à la classe de problèmes d'équilibrage de lignes d'assemblage qui représentent les problèmes les plus proches de ceux qui nous intéressent.

Chapitre 3

Problèmes d'équilibrage des lignes de production

Pour répondre concrètement aux problèmes qui se posent dans la production d'une façon générale et dans l'usinage en particulier, il est indispensable d'admettre des hypothèses de travail. Ces hypothèses restreignent la part de réalité du problème traité dans le but de réduire sa complexité. Elles sont nécessaires pour apporter des réponses précises à des problèmes difficiles à aborder. Une bonne illustration de cette approche est le problème de l'équilibrage des lignes d'assemblage (ALB) qui a été largement étudié dans la littérature. Il en dérive plusieurs problèmes : le simple, noté SALBP, pour *Simple Assembly Line Balancing Problem*, et les généralisés notés GALBP, pour *Generalized Assembly Line Balancing Problem*. Ces problèmes sont moins complexes que celui que nous traitons, néanmoins nous avons fait le choix de les aborder car ils ont de nombreux points en commun avec les problèmes de configuration de lignes d'usinage que nous étudions dans ce mémoire. Nous les décrivons de façon formelle indiquons leurs hypothèses dans cette section. Nous présentons également quelques méthodes d'optimisation qui ont été employées pour les résoudre.

3.1 Le SALBP et ses variantes

Une ligne d'assemblage consiste en une série de stations, chacune effectuant un ensemble d'opérations[1]. Les opérations sont caractérisées par leur temps d'exécution et elles sont le plus souvent reliées par des contraintes de précédence. Les opérations d'une même station sont exécutées de façon séquentielle. Il en résulte que le temps de travail d'une station est égal à la somme des temps d'exécution de ses opérations. Le temps de cycle impose une cadence aux stations car celles-ci ne doivent pas avoir un temps de travail qui lui est supérieur. Le temps mort d'une station est défini comme étant la différence entre le temps de cycle et son temps de travail. L'équilibrage d'une ligne d'assemblage revient à trouver une affectation de l'ensemble des opérations telle que les contraintes de précédence entre les opérations soient respectées et que le temps mort total de la ligne soit minimal.

[1]Le mot *tâche* est également utilisé pour désigner une opération.

La littérature est particulièrement abondante en travaux pour le SALBP, elle l'est davantage pour sa variante SALBP-1. Nous ne rapporterons, dans ce qui suit, que les travaux qui ont un lien méthodologique avec nos travaux ou qui emploient des techniques proches de celles que nous avons étudiées.

Les hypothèses générales du SALBP, qui ont été posées par Baybars [Bay86], sont les suivantes :

1. les temps d'exécution des opérations sont déterministes,
2. les temps d'exécution des opérations sont indépendants de la station sur laquelle elles sont effectuées,
3. toute opération peut être exécutée sur toute station,
4. toutes les opérations doivent être exécutées,
5. un ordre partiel entre les opérations doit être respecté,
6. une opération est exécutée complètement sur une seule station, c'est-à-dire qu'il ne peut y avoir de chevauchement sur une autre station,
7. la ligne est constituée d'une série de stations et le mode de fonctionnement est synchrone,
8. les stations sont visitées dans un ordre donné,
9. un seul type de produit est considéré,
10. selon les variantes du SALBP soit le temps de cycle T_0 est donné soit le nombre de stations m est donné.

Il existe plusieurs variantes de SALBP, elles se différencient notamment par la variation de la dernière hypothèse mais également par leur objectif. La sous-section qui suit est consacrée à décrire ces différentes types en précisant à chaque fois leurs particularités.

Le tableau 3.1 [SB06] synthétise les différentes variantes du SALBP. Le problème SALBP-F est un problème de décision (le F désignant faisabilité). Dans ce cas, le problème revient à trouver une solution réalisable qui respecte les données, en l'occurrence le temps de cycle et le nombre de stations. Contrairement au SALBP-F, les variantes SALBP1 et SLABP2 sont des problèmes d'optimisation. Le premier problème a comme objectif de minimiser le nombre de stations en respectant un temps de cycle donné et fixe, alors que le second, à l'inverse, doit minimiser le temps de cycle en respectant un nombre de stations donné et fixe. Le SALBP-E est plus général car il a pour objectif de maximiser l'efficacité $E = \sum_{i=1}^{n} t_i / mT_0$ (tel que n est le nombre d'opérations et t_i est le temps d'exécution de l'opération i). Du fait que $\sum_{i=1}^{n} t_i$ est fixe, l'objectif revient à minimiser le produit du temps de cycle et du nombre de stations en même temps soit à minimiser mT_0.

Plusieurs états de l'art détaillés sur le SALBP sont proposés dans [BSW73, Bay86, GG89, ES98, RDDB02] et plus récemment [SB06].

3.1.1 Complexité

Du fait que la variante du SALBP-F traitant de la faisabilité du SALBP est, elle-même, un problème NP-complet, les problèmes d'optimisation correspondants, c'est-à-dire SALBP-

	Temps de cycle	
Nombre de stations	Donné	À minimiser
Donné	SALBP-F	SALBP-2
À minimiser	SALBP-1	SALBP-E

TAB. 3.1 – Les variantes du SALBP

1 et SALBP-2, le sont également, car ces derniers peuvent être résolus en parcourant les solutions du SALBP-F [Kar72, WM86, SB06]. La forte combinatoire du problème a conduit à la recherche de bornes inférieures afin de réduire l'espace de recherche. Comme le montrent les travaux faits dans ce sens que nous citons ci-dessous, les bornes inférieures sont essentiellement obtenues en résolvant des relaxations du problème original en un problème plus facile. Dans ce qui suit, nous rapportons les différentes bornes inférieures proposées pour le SALBP-1, c'est-à-dire pour le nombre de stations (minimiser le temps mort pour le SALBP-1 est équivalent à la minimisation du nombre de stations).

3.1.2 Bornes inférieures pour le SALBP-1

Différentes bornes ont été proposées pour la fonction objectif du SALBP-1. Elles se distinguent par leur complexité, celles qui ont une complexité linéaire permettent d'obtenir une estimation rapide mais souvent imprécise. Comme nous l'avons déjà évoqué précédemment, plus d'efforts sont déployés dans les calculs, meilleure est la qualité de la borne. Ce principe est confirmé dans le cas du SALBP-1 ainsi les bornes les plus élaborées fournissent une meilleure approximation de l'objectif, contre-partie d'un plus grand temps de calcul.

Méthode constructive

Les bornes suivantes sont les plus intéressantes du point de vue du temps de calcul car elles peuvent être obtenues avec un algorithme de complexité linéaire (c'est-à-dire en $\mathcal{O}(n)$).

Notons l'ensemble des opérations $\mathbf{N} = \{1, \ldots, n\}$, pour toute opération $i \in \mathbf{N}$, son temps d'exécution est donné par t_i. La borne donnée par (3.2) est appelée *nombre de stations minimum théorique* [Bay86]. Elle dérive du temps de séjour sur la ligne qui doit être au moins égal à la somme des temps d'exécution de toutes les opérations $i \in \mathbf{N}$, c'est-à-dire :

$$mT_0 \geq \sum_{i=1}^{|\mathbf{N}|} t_i \tag{3.1}$$

avec m désignant le nombre de stations et T_0 le temps de cycle.

$$LB_1 = \left\lceil \sum_{i=1}^{|\mathbf{N}|} \frac{t_i}{T_0} \right\rceil \tag{3.2}$$

Tel que $\lceil a \rceil$ est le plus petit entier supérieur ou égal à a.

La borne LB_1 a une complexité en $\mathcal{O}(n)$, ce qui correspond, en pratique, au parcours des $|\mathbf{N}|$ temps d'exécution des opérations.

Une deuxième borne est proposée par Johnson [Joh88], elle est obtenue en considérant les opérations avec un certain temps d'exécution. Plus exactement, si on considère la proportion de temps correspondant à l'occupation d'une opération dans une station, *i.e.* $p_i = t_i/T_0$, alors il est possible de déduire que les opérations ayant une proportion supérieure à 0,5 ne peuvent pas partager une même station, elles définissent alors une borne inférieure sur le nombre de stations. Celles qui ont une proportion égale à 0,5 se voient attribuer la moitié d'une station.

$$LB_2 = \big|\{i|\frac{1}{2} < p_i \leq 1\}\big| + \Big\lceil \frac{1}{2}.\big|\{i|p_i = \frac{1}{2}\}\big| \Big\rceil \tag{3.3}$$

Il est à noter que la borne LB_2 est moins performante que la précédente lorsque le problème comporte de nombreuses opérations ayant des proportions strictement inférieures à 0,5, toutefois dans le cas inverse c'est LB_2 qui fournie une meilleure valeur.

Le raisonnement fait précédemment pour les proportions égales à la moitié de la charge d'une station peut être appliqué aux proportions de tiers. Ainsi, les opérations ayant une proportion supérieure à 2/3 déterminent des stations différentes. Les opérations avec un p_i de 1/3 et 2/3 peuvent partager une même station et leur est attribué un coefficient de 1/3 et 2/3 respectivement. Les opérations avec $\frac{1}{3} \leq p_i \leq \frac{2}{3}$ peuvent au moins partager une même station, ils leur est attribué un coefficient de 1/2. Ainsi la borne est égale à :

$$LB_3 = \big|\{i|\frac{2}{3} \leq p_i \leq 1\}\big| + \Big\lceil \frac{2}{3}\big|\{i|p_i = \frac{2}{3}\}\big| + \frac{1}{2}\big|\{i|\frac{1}{3} < p_i < \frac{2}{3}\}\big| + \frac{1}{3}\big|\{i|p_i = \frac{1}{3}\big| \Big\rceil \tag{3.4}$$

Relaxation à un problème d'ordonnancement à une machine

Une borne plus élaborée est obtenue en relâchant le SALBP-1 à un problème d'ordonnancement à une seule machine [Joh88]. Les opérations deviennent les travaux (jobs) qui doivent passer sur la machine avec des temps d'exécution égaux à $p_j = t_j/T_0$ pour chaque opération j. Toutes les opérations doivent passer sur la machine de façon séquentielle. Après chaque travail j, un temps, noté n_j et appelé *tail*, est nécessaire pour achever le reste. L'objectif est de trouver la séquence qui minimise le temps d'exécution total, c'est-à-dire le *makespan*. La solution optimale pour un tel problème est obtenue en exécutant les travaux dans un ordre décroissant de n_j. Pour le SALBP-1, pour chaque opération j, la valeur de n_j est une borne inférieure sur le nombre de stations nécessaires pour exécuter toutes ses opérations successeurs. Les n_j sont calculés de façon récursive en sens inverse de l'ordre topologique, tel que l'ordre topologique est déduit des rangs des opérations dans le graphe de précédence (il faut ajouter deux nœuds fictifs une source 0 et un puits $n+1$). Ainsi, pour obtenir n_j d'une opération j ayant l'ensemble F^* de successeurs, il est nécessaire de calculer l'ensemble des tails n_i de ces successeurs i démarrant par le dernier nœud $n+1$ qui n'a pas de successeurs. Nous notons $< h_1, \ldots, h_r], \quad r \in F^*$ la liste ordonné des successeurs de j.

$n_j = maximum\{p_{h_1} + n_{h_1}, p_{h_1} + p_{h_2} + n_{h_2}, ..., p_{h_1} + ... + p_{h_n} + n_{h_n}\}$

La borne n_j, pour l'opération j, n'est pas forcément entière. Elle peut également être obtenue au moyen des bornes décrites précédemment, c'est-à-dire LB_1, LB_2 ou LB_3 sans les arrondir.

Ainsi, une borne inférieure sur le nombre de stations de la ligne est donnée par $\lceil n_0 \rceil$, c'est-à-dire le tail du nœud fictif n'ayant aucun prédécesseur.

Le résultat précédent peut être affiné davantage en calculant, pour chaque opération j, une borne inférieure (dite *head* et notée a_j) sur le nombre de stations nécessaires à l'execution des opérations prédécesseurs de j. Par analogie au calcul des n_j le même raisonnement est fait pour l'obtention des a_j. Celles-ci sont calculées de façon récursive en commençant par a_0 remontant jusqu'à a_{n+1}. De la même façon que pour $\lceil n_0 \rceil$, $\lceil a_{n+1} \rceil$ est une borne sur le nombre de stations.

Il est à signaler que $Z = \max_{j=0,\ldots,n+1}\{a_j + p_j + n_j\}$ est aussi une borne inférieure sur le *makespan* du problème d'ordonnancement à une seule machine utilisant les *heads* et les *tails* [Car82], alors :

$$LB_4 = \max\{\lceil n_0 \rceil, \lceil a_{n+1} \rceil, Z\} \tag{3.5}$$

La complexité des bornes LB_1,LB_2,LB_3 est linéaire c'est-à -dire en $\mathcal{O}(n)$. Par contre, le calcul de LB_4 a au moins une complexité en $\mathcal{O}(n^2)$, selon la borne utilisée pour le calcul des limites a_j et n_j, $j = 1, \ldots, n$.

Amélioration incrémentale

Par opposition aux calculs des bornes décrites précédemment, la borne LB_5 est basée sur un raisonnement incrémental sur la valeur de m (le nombre de stations). Cette valeur peut être initialisée avec la valeur de LB_1, par exemple. L'idée d'amélioration des itérations réside dans la détection d'une ou plusieurs infaisabilités qui seraient engendrées par la valeur courante de m, si c'est le cas la borne est incrémentée et le processus est réitéré, autrement le processus s'arrête car il n'est plus possible d'améliorer la borne.

La borne LB_5, proposée par [SB87], utilise la notion de station au plus tôt, notée E_j, et de station au plus tard notée L_j, pour chaque opération j telle que la station au plus tard est calculée en fonction de m. Cependant, afin de distinguer les stations au plus tard liées à chaque valeur de m, $L_j(m)$ est utilisé pour désigner la station au plus tard de l'opération j à l'itération correspondante à la valeur m. Pour obtenir une solution réalisable, il est nécessaire que la condition $E_j \leq L_j(m)$ soit vérifiée pour toutes les opérations, autrement l'opération correspondante ne pourra être effectuée avec le nombre de stations courant. Ainsi :

$$LB_5 = min\{m | L_{j(m)} \geq E_j\}$$

Cette dernière expression peut être affinée en utilisant les limites a_j et n_j introduites pour le calcul de LB_4. Ainsi, les E_j et $L_{j(m)}$ sont données par (3.6) et (3.7) :

$$E_j = \lceil a_j + p_j \rceil, \quad \forall j = 1, \ldots, n \tag{3.6}$$

$$L_{j(m)} = m + 1 - \lceil p_j + n_j \rceil, \quad \forall j = 1, \ldots, n \tag{3.7}$$

Dans ce cas, le temps de calcul pour obtenir la borne LB_5 dépend du temps de calcul des limites a_j et n_j, $j = 1, \ldots, n$.

3.2 Méthodes exactes pour le SALBP-1

Deux familles d'approches de résolution exactes ont été proposées pour le SALBP, la première est basée sur la programmation dynamique [HKS63] et la deuxième est basée sur une procédure par séparation et évaluation. Nous présentons brièvement la première approche et développons plus en détails la seconde car celle-ci a montré sa supériorité tant sur le plan du temps de calcul que sur le plan de l'espace mémoire requis. Nous présentons donc trois procédures : Fable [Joh88], Eureka [Hof92] et Salome-1 [SK97]. Ces procédures mettent en pratique un large éventail des techniques de réduction et de calcul de bornes les plus efficaces qui ont été développées pour le SALBP-1.

3.2.1 Programmation dynamique

Les approches basées sur la programmation dynamique se déclinent, selon leur orientation, en deux catégories, elles peuvent être orientées stations ou opérations (tâches). Par exemple, la procédure de Jackson [Jac56] est orientée station. De plus, les nœuds correspondent à différents états représentant chacun un chargement complet d'une station. L'affectation de l'ensemble des opérations d'une station est représentée par un arc valué avec le temps mort de la station correspondante. Pour résoudre le SALBP-1, il suffit de résoudre le problème de plus court chemin dans le graphe construit, voir également [GN64, Man67, Kle63].

Concernant les approches orientées opérations où les états correspondent à l'affectation d'une seule opération, nous citons celle proposée par Baker & Scharge [BS78] et Kao & Queyranne [KQ82]. La dernière est une amélioration de celle de Held, Karp et Shareshian [HKS63]. L'inconvénient majeur est le large espace mémoire requis, et ce malgré les efforts d'améliorations apportés pour le réduire.

3.2.2 Fable

La procédure *Fable* met en place un schéma de branchement *orienté tâche*, c'est-à-dire qu'un branchement correspond à l'affectation d'une seule opération à la fois. Un procédé itératif est enclenché pour toutes les opérations qui peuvent être assignées au vu de la solution courante. Les sous-problèmes sont construits en ajoutant, à chaque fois, une seule opération parmi celles qui peut être affectées à la station courante. Si le temps restant sur cette station ne permet pas une telle affectation alors la station est chargée au maximum et l'algorithme ouvre (ajoute) une nouvelle station qui sera désignée à son tour *station courante*. Le branchement se fait en profondeur d'abord, *Depth First Search*, les opérations candidates

sont triées selon l'ordre topologique. Dès qu'une solution réalisable est atteinte, un mécanisme de backtracking est enclenché remontant jusqu'à un point où d'autres branchements sont encore possibles en choisissant toujours de telle façon que l'ordre topologique entre opérations soit respecté.

En termes de bornes inférieures, Fable employe les bornes LB_1,LB_2,LB_3 et LB_4, décrites dans les sections précédentes. Les trois premières bornes sont utilisées à chaque nœud, alors que LB_4 n'est utilisée qu'au nœud racine car son calcul nécessite plus de temps en raison de sa complexité. Des règles de dominance sont également intégrées dans le schéma de branchement et permettent de couper des branches non prometteuses. En outre, une règle d'incrémentation des temps opératoires est employée pour les opérations qui ne peuvent être combinées avec les autres : le principe est d'arrondir leur temps opératoire à la valeur du temps de cycle, ce qui permet d'améliorer la valeur de la borne inférieure.

3.2.3 Eureka

L'arbre de recherche est orienté station, c'est-à-dire chaque branchement correspond à un chargement total d'une station. Connue aussi sous le nom de méthodes de borne inférieure, Eureka est un processus répétitif qui tente de trouver une solution n'excédant pas la meilleure valeur théorique du nombre de stations (LB), c'est-à-dire la valeur de la borne inférieure. Néanmoins, si aucune solution n'est trouvée alors la valeur de LB est incrémentée et le processus est réitéré. Tous les nœuds qui correspondent à des solutions partielles dans lesquelles le temps mort est supérieur à la valeur $LB \times T_0 - \sum_{i \in A} t_i$, tel que A est l'ensemble des opérations qui ont déjà été affectées, sont élagués (car la solution correspondante contient plus que LB stations). Lorsque la recherche est fructueuse, une solution réalisable est obtenue, elle correspond forcément à une solution optimale car elle contient LB stations et la condition d'arrêt est satisfaite. L'algorithme fonctionne en deux étapes, la première construit les solution avançant de la première station vers la *n-ème* ; quant à la seconde, elle construit en sens inverse les stations à partir de la dernière remontant jusqu'à la première, et ce en utilisant le graphe de précédence inversé. En fait, un temps limité est accordé à la procédure en avant pour trouver une solution et lorsque celle-ci échoue la procédure cherchant dans le sens inverse est enclenchée pour le même intervalle de temps. Si une des procédures trouve une solution, l'algorithme est arrêté car cette dernière est la solution optimale.

3.2.4 Comparaison entre Eureka et Fable

La stratégie de branchement orienté tâche de Fable permet, contrairement à Eureka, d'éviter de construire systématiquement des solutions complètes. Toutefois, les bornes sont plus efficaces lorsque les stations sont chargées au maximum, en outre, le temps mort de la station ne peut être estimé qu'à ce niveau et donc ce sont à ces points précisément qu'un élagage potentiel est plus probable. Fable peut être renforcé dans ce sens, en calculant la borne sur le nombre de stations au niveau des nœuds où le chargement des stations est maximal (c'est-à-dire aux points où les stations sont fermées).

Par ailleurs, le calcul des bornes est moins fréquent dans le cas d'Eureka. En même temps, l'utilisation de LB_1 qui est propre à Eureka n'est pas le meilleur choix car la valeur de la borne inférieure peut être plus précise en employant, par exemple, la borne LB_4 [Sch99].

Ainsi, ces deux méthodes présentent des points forts mais aussi des faiblesses, elles peuvent être améliorées et combinées de façon à exploiter leur avantages. C'est dans cette perspective que s'inscrit la méthode SALOME-1, proposée dans [SK97].

3.2.5 SALOME-1

Le schéma de branchement de SALOME-1 est orienté station, comme dans Eureka, car la borne inférieure est plus efficace dans ce cas, et la stratégie de l'exploration de l'arbre est en profondeur d'abord afin d'obtenir des solutions faisables au plus tôt. L'idée de l'amélioration apportée par SALOME-1 est de remplacer le processus itératif d'Eureka qui construit plusieurs arbres par un autre n'en développant qu'un seul. L'effort est centralisé dans la construction de cet arbre en mettant en place une borne inférieure de bonne qualité. Celle-ci sera calculée à chaque nœud au lieu de l'employer juste à la racine comme dans Eureka. Notons que cette méthode a été adaptée avec succès pour un problème de bin-packing [SK97].

Pour des problèmes de plus grande taille, des méthodes approchées semblent plus appropriées pour une résolution efficace. Pour ce faire, une approche basée sur une procédure par séparation et évaluation tronquée est souvent employée dans ce but. Le critère d'arrêt peut être soit un temps limite à ne pas dépasser soit un nombre de nœuds maximum [Hof63, GP78, HM78]. Quelques autres méthodes approchées sont présentées dans la section suivante.

3.3 Méthodes approchées pour le SALBP-1

3.3.1 Méthodes constructives

Il existe un nombre important de travaux proposant des heuristiques pour résoudre le SALBP-1. Une revue et comparaison de ces heuristiques est rapportée dans [TPG86]. Dans ce qui suit nous avons choisi de présenter RPW pour *Ranked Positional Weight* qui est une heuristique basée sur une simple règle de choix (une seule solution est construite, c'est-à-dire une heuristique mono-passage) puis nous rapportons une description de COMSOAL pour *COmputer Method of Sequencing Operations for Assembly Lines* qui, à l'inverse de la précédente, exploite un choix aléatoire en générant plusieurs solutions et en gardant la meilleure (une heuristique multi-passage).

RPW

Cette procédure est introduite dans [HB61]. Il s'agit d'un algorithme glouton basé sur une règle de priorité statique car le choix des opérations à sélectionner est indépendant du

processus de construction des solutions. Cette règle heuristique s'appuie sur un calcul de poids pour chaque opération, ce calcul n'est effectué qu'une seule fois et les valeurs des poids obtenues restent inchangées pendant le déroulement de l'algorithme. Le poids d'une opération est obtenu en additionnant son temps d'exécution avec la somme de tous les temps opératoires de ses successeurs dans le graphe de précédence. Une liste est obtenue en triant l'ensemble des opérations par ordre décroissant de leur poids. Le processus de construction propose d'affecter sur la station courante d'abord l'opération en tête de liste, c'est-à-dire celle qui a le plus grand poids, à condition que ce qui suit soit vérifié :

1. le temps restant sur la station courante est supérieur ou égal au temps nécessaire à l'exécution de l'opération candidate,
2. tous les prédécesseurs de cette opération ont été déjà affectés.

Si l'opération en tête de liste ne respecte pas ce choix alors c'est l'opération suivante de la liste qui est considérée, etc. Toutefois, si aucune opération de la liste ne peut être affectée à la station courante celle-ci est fermée et une nouvelle station est ouverte, qui devient la station courante. Ensuite, la liste des opérations candidates est mise à jour en supprimant celles qui ont été affectées. Le processus est réitéré jusqu'à ce que toutes les opérations soient assignées.

COMSOAL

Arcus [Arc66] propose une version stochastique d'une heuristique multi-passage, la procédure génère plusieurs solutions complètes jusqu'à ce qu'un certain nombre soit atteint. À chaque itération, une opération à affecter est choisie aléatoirement à partir de la liste des opérations disponibles, c'est-à-dire celles qui ont leurs prédécesseurs déjà affectés et dont le temps d'exécution est inférieur au temps restant sur la station courante. Le processus est réitéré jusqu'à l'obtention d'une solution complète. À l'évidence, plus le nombre de solutions construites est important, plus élevées sont les chances de trouver l'optimum (ou une solution proche).

3.3.2 Méthodes incrémentales

Ces méthodes sont basées sur l'amélioration successives des solutions (celles-ci peuvent être initialement obtenues avec des méthodes constructives). Par exemple, les algorithmes génétiques manipulent une population de solutions grâce à plusieurs opérateurs de type croisement, sélection et mutation. Leur application fait apparaître quelques difficultés telles que le codage des solutions, le lissage de la fonction fitness et la réparation des solutions rendues irréalisables après l'application de croisements. Nous faisons référence aux travaux de [KKK96, Fal93, PCPV03].

Une autre metaheuristique pour le SALBP est proposée dans [LRS06], la procédure est basée sur la recherche tabou avec deux différences majeures par rapport aux travaux antérieurs de [Chi98]. La première réside dans le fait d'employer deux techniques de recherche locales, qui sont complémentaires, pour l'intensification et la diversification. De plus, des

redéfinitions de l'espace de recherche et de l'objectif ont été suggérées afin d'explorer les solutions infaisables ne respectant pas les contraintes de temps de cycle.

Dans ce qui suit, nous rapportons les travaux qui ont traité de problèmes plus larges, tel que l'équilibrage avec le choix d'équipement ou le CALBP.

3.4 Généralisations (GALBP)

Les travaux proposés pour le SALBP ont servi de base pour l'étude de plusieurs extensions. Ces généralisations sont englobées sous la notation de *GALBP* pour *Generalized Assembly Line Balancing Problem*. Une de ces généralisation relaxe la deuxième hypothèse de SALBP-1 (voir 3.1) considérant les temps des opérations comme étant indépendant des stations, ce qui implique que l'ensemble de l'équipement est le même. Dans la pratique, c'est rarement le cas, les temps opératoires sont en réalité directement liés au type d'équipement utilisé.

3.4.1 Sélection d'équipement

Nous rapportons, ici, les travaux qui ont abordé les problèmes d'équilibrage des lignes d'assemblages en s'intéressant simultanément à la sélection d'équipements. Ces travaux sont présentés par ordre chronologique.

[GL83] considèrent le cas mono-produit et s'intéressent à la conception des lignes d'assemblage avec la sélection d'équipement en considérant un ordre complet entre les opérations.

[PDK83] étudient une version étendue du SALBP considérant plusieurs alternatives de processus pour une ligne d'assemblage manuelle[2]. Du fait que la ligne est manuelle, les opérations peuvent être effectuées indifféremment sur toutes les stations. Chaque alternative est relative à un ensemble d'opérations et demande des équipements spécialisés pour cet ensemble d'opérations ; ces équipements supplémentaires peuvent alors être ajoutés à l'équipement déjà en place dans la station correspondante. Les auteurs posent le problème comme suit : étant donné un coût induit par l'utilisation d'équipements supplémentaires est-il intéressant de les employer pour réduire le temps d'exécution des opérations ? Pour y répondre, une procédure de séparation et évaluation est construite où un SALBP-1 est résolue à chaque nœud. Cette méthode n'est efficace que pour un petit nombre d'équipements possibles.

[GHR88] considèrent le problème de conception des lignes s'assemblage en présence de plusieurs types d'équipements pour le cas multi-produit. Des hypothèses assumant un ordre complet des opérations pour le même produit et imposant une grande similarité entre les produits ont permis de réduire de façon considérable le nombre de stations possibles. Plus précisément, ce nombre est ramené à N^k où N est le nombre total d'opérations et k a une valeur proche de 2. L'algorithme proposé énumère toutes les possibilités d'équipements pour les stations, pour n'en retenir que les meilleurs.

[2]Par ligne d'assemblage manuelle nous désignons une ligne qui fait intervenir uniquement des opérateurs humains.

Dans [BT00], une affectation des opérations avec une sélection d'équipement pour la conception des lignes d'assemblage flexibles est proposée. En présence de plusieurs types d'équipements disponibles, il faut en choisir un seul pour chaque station et y affecter en même temps des opérations. Chaque équipement a un coût spécifique et le temps d'exécution d'une opération dépend du type d'équipements choisi, c'est-à-dire qu'il y a des équipements qui sont plus performants pour effectuer certaines opérations mais qui seront plus chers et/ou moins performantes pour d'autres opérations. Quelque soit l'équipement choisi, les opérations de la même station sont effectuées en séquence. L'objectif est de minimiser le coût de la ligne qui est composé du coût de l'équipement choisi, et ce en respectant le temps de cycle objectif (déduit du taux de production visé). Les auteurs proposent de modéliser le problème à l'aide d'un programme linéaire en variables binaires et de le résoudre avec une procédure de séparation et d'évaluation. Les auteurs proposent une borne inférieure basée sur la relaxation des contraintes de précédence et d'intégrité des variables. Néanmoins l'algorithme n'est capable de résoudre que des problèmes de taille modeste dans un lapse de temps fixé. Pour les problèmes de plus grande taille, ils proposent un algorithme approché obtenu en tronquant la procédure par séparation et évaluation.

Une version étendue du précédent algorithme est présentée dans [BR03]. Le problème traité prend en compte des stations parallèles.

3.4.2 Équilibrage orienté coût (CALBP)

Dans [Ame00a], l'auteur présente le problème CALBP, pour *Cost oriented Assembly Line Balancing Problem*, pour lequel il propose un état de l'art dans [Ame00b]. Dans ce problème d'optimisation il ne s'agit plus de minimiser le nombre de stations mais plutôt le coût engendré par la configuration de la ligne. Plus exactement, l'objectif est posé comme la minimisation du coût salarial lié aux niveaux de qualification des ouvriers impliquant des salaires différents. L'idée proposée est de regrouper les opérations par rapport à un niveau de qualification proche afin qu'un opérateur ayant ce niveau les effectue. En d'autres termes, il faut que les opérateurs hautement qualifiés n'interviennent que pour les opérations les plus difficiles à réaliser, le regroupement des opérations de même niveau permet de centraliser leur intervention. Du fait de la nature de l'objectif, la règle appliquée pour le SALBP tendant à minimiser le nombre de stations ne mène plus forcément à la solution optimale. En effet, la minimisation du nombre de stations peut écarter des solutions optimales pour le CALBP. En particulier, certaines solutions peuvent comporter un plus grand nombre de stations mais revenir moins cher que d'autres configurations comportant moins de stations dans lesquelles les qualifications ont été mal affectées. Une méthode exacte et une borne inférieure pour le nombre de stations et pour le coût sont proposées dans [Ame00a] et une méthode heuristique est développée dans [Ame01].

3.5 Problèmes d'équilibrage des lignes d'usinage

À présent, nous allons présenté les travaux concernant l'équilibrage des lignes d'usinage. La littérature est beaucoup moins abondante dans ce domaine malgré son importance. Pour l'usinage, les objectifs a atteindre ne sont pas forcément les mêmes que pour les lignes d'assemblage. En effet, lorsque pour l'assemblage manuel il est important d'équilibrer la charge de travail entre les postes en raison de la présence d'opérateurs humains, pour les lignes d'usinage la préoccupation n'est plus à lisser la charge car les lignes sont complètement automatisées. Par contre, il est essentiel de réduire le coût d'investissement pour ce type de lignes pour les raisons que nous avons déjà évoquées (voir section 1.4).

Dans la suite, nous passons en revue les travaux proposés dans le cadre de l'usinage en terminant par ceux qui ont été effectués par notre équipe de recherche.

Dans [Sza97], un problème d'optimisation de processus d'usinage est étudié. L'auteur se limite à la formulation du SALBP-1 et applique la méthode de recherche du plus court chemin dans un graphe spécialement construit. L'originalité du travail réside dans le fait que l'auteur considère pour chaque opération plusieurs états représentés par le couple : opération et positionnement de la pièce. L'approche intègre également les régimes d'usinage (vitesse de coupe) dans les variables de décision.

Les travaux de Masood [Mas06] proposent une étude de cas dans le cadre de l'équilibrage d'une ligne de transfert pour la fabrication de blocs de cylindres. L'objectif est de réduire le temps de cycle et d'augmenter le taux d'utilisation de la ligne. Pour ce faire, l'auteur utilise une approche par simulation qui tient compte du changement d'outils traitant des problèmes de 10 outils et 16 opérations.

Les travaux de [YUA02] se placent dans un environnement de RMS. Les auteurs y proposent d'optimiser des lignes d'usinage fabriquant des produits modulaires. Par production modulaire, ils entendent la production de plusieurs instances de modules, qu'ils peuvent combiner par la suite pour obtenir des produits finaux. Cette modularité permet d'atteindre une grande variété de produits finaux lorsque le nombre d'instances par module est suffisamment significatif. Ainsi, une instance de chaque module peut être combiné avec d'autres instances d'autres modules pour composer un produit final. Une ligne est utilisée pour la production de chaque module, il y a donc autant de lignes que de modules. Chaque ligne est initialement configurée pour la fabrication d'une instance donnée et le passage à la production d'une autre instance devient possible suite à une reconfiguration de cette ligne. Le problème posé est alors de trouver la meilleure granularité possible afin d'avoir la diversification voulue à moindre coût (le coût étant engendré par les reconfigurations des lignes). En fait, plus les instances d'un module sont différentes plus la reconfiguration sera coûteuse car les modifications apportées seront plus lourdes. D'un autre côté, si les instances sont choisies de façon à minimiser le coût de reconfiguration, le nombre de ces instances sera élevé implicant une augmentation du nombre de reconfigurations et du coût total.

Il est également intéressant de citer les travaux de [TYHWK03] qui proposent d'appliquer les algorithmes génétiques pour résoudre simultanément le problème de l'équilibrage d'une ligne multi-produit et celui de la sélection des machines et des stocks tampons. L'objectif

considéré est la maximisation de la productivité de la ligne.

3.6 Optimisation des lignes de transfert

Nous rapportons dans cette section les travaux qui ont été effectués pour la conception préliminaire des lignes de transfert dans le cadre de la thèse de [Fin04]. Cette thèse s'est intéressée à la structuration de lignes de transfert en termes de définition de blocs d'opérations et de leur affectation aux stations. Ce problème est noté TLBP pour *Transfer Line Balancing Line Problem*. Les opérations d'un bloc sont exécutées en parallèle par la tête multi-broche (unité) correspondante. Le problème d'optimisation posé revient à trouver les regroupements des opérations en blocs et une affectation de ces blocs aux stations.

Plusieurs contraintes ont été introduites :

- certaines opérations ne doivent pas être exécutées sur une même station,
- certaines opérations ne peuvent pas être affectées dans un même bloc,
- certaines opérations doivent être affectées dans une même station,
- certaines opérations doivent être exécutées dans le même bloc.

D'autres contraintes technologiques ont également été prises en compte : elles consistent à limiter le nombre d'opérations par bloc, le nombre de blocs par station et le nombre de stations sur la ligne.

Nous rapportons dans ce qui suit les différences essentielles entre le problème étudié dans [Fin04] et celui que nous traitons dans ce mémoire :

Dans les travaux de [Fin04] tous les regroupements possibles des opérations sont envisagés et représentent des blocs potentiels, tandis que dans notre cas seuls les regroupements des opérations pour lesquels il existe des têtes d'usinage qui peuvent les effectuer sont considérés. Autrement dit, dans le premier cas, les blocs sont construits durant le processus d'optimisation et la solution fournit les regroupements des opérations correspondant aux têtes qu'il faudra acquérir (ou concevoir et fabriquer). Se posent alors les problèmes de la disponibilité (ou faisabilité) de ces têtes d'usinage et de leur coût. En d'autres termes, les solutions fournies ne permettent pas toujours de réaliser les structures de lignes, voire pire, elles peuvent engendrer des coûts supplémentaires inutilement. Pour y remédier, nous avons proposé de construire d'abord l'ensemble des têtes d'usinages faisables qui va constituer la base de la procédure de recherche des configurations. Cet ensemble est une donnée de départ, toutes les structures proposées deviennent réalisables et restent en adéquation avec la disponibilité du marché. Les travaux de [Fin04] peuvent s'inscrire dans une démarche antérieure à l'étude qui nous intéresse dans cette thèse pour nous fournir les unités d'usinage.

Une autre différence importante réside dans la définition de l'objectif à minimiser. Dans [Fin04], l'objectif est de minimiser le nombre de stations et têtes d'usinage, considérant une somme pondérée de ces derniers. Plus exactement, le même coût est attribué à toute tête qui est construite et un coût propre à la création d'une station est intégré. Les pondérations

représentent des poids induisant des coûts relatifs des têtes par rapport aux stations. L'ajustement de ces paramètres permet de définir une proportion de coût d'une tête par rapport à celui d'une station. Du fait que les têtes d'usinage sont construites durant le processus d'optimisation, il devient impossible d'estimer leur coût à cette étape. Dans cette thèse, étant donné l'ensemble des unités d'usinage disponibles nous proposons d'affiner l'optimisation en considérant le coût total comprenant le coût des unités et des stations.

Dans ces travaux, c'est le mode séquentiel pour l'activation des têtes d'usinage qui est considéré. Alors que dans cette thèse, nous abordons tous les cas possibles, à savoir : le cas parallèle où les têtes sont exécutés de façon simultanée et le cas mixte qui est une généralisation du mode séquentiel et parallèle.

Pour le problème traité dans [Fin04] de nombreuses approches ont été proposées. Entre autres, une approche basée sur la programmation en variables mixtes [DFG+06b]. Les auteurs ont utilisé Xpress MP et ensuite ILOG Cplex, pour implémenter et résoudre les modèles PLVM proposés. Dans [DFG+05] deux heuristiques ont été suggérées en se basant sur la méthode COMSOAL. Dans [DFG+06a] une autre heuristique est suggérée par les auteurs qui décomposent le problème en plusieurs sous-problèmes où chaque sous-problème est résolu de manière exacte.

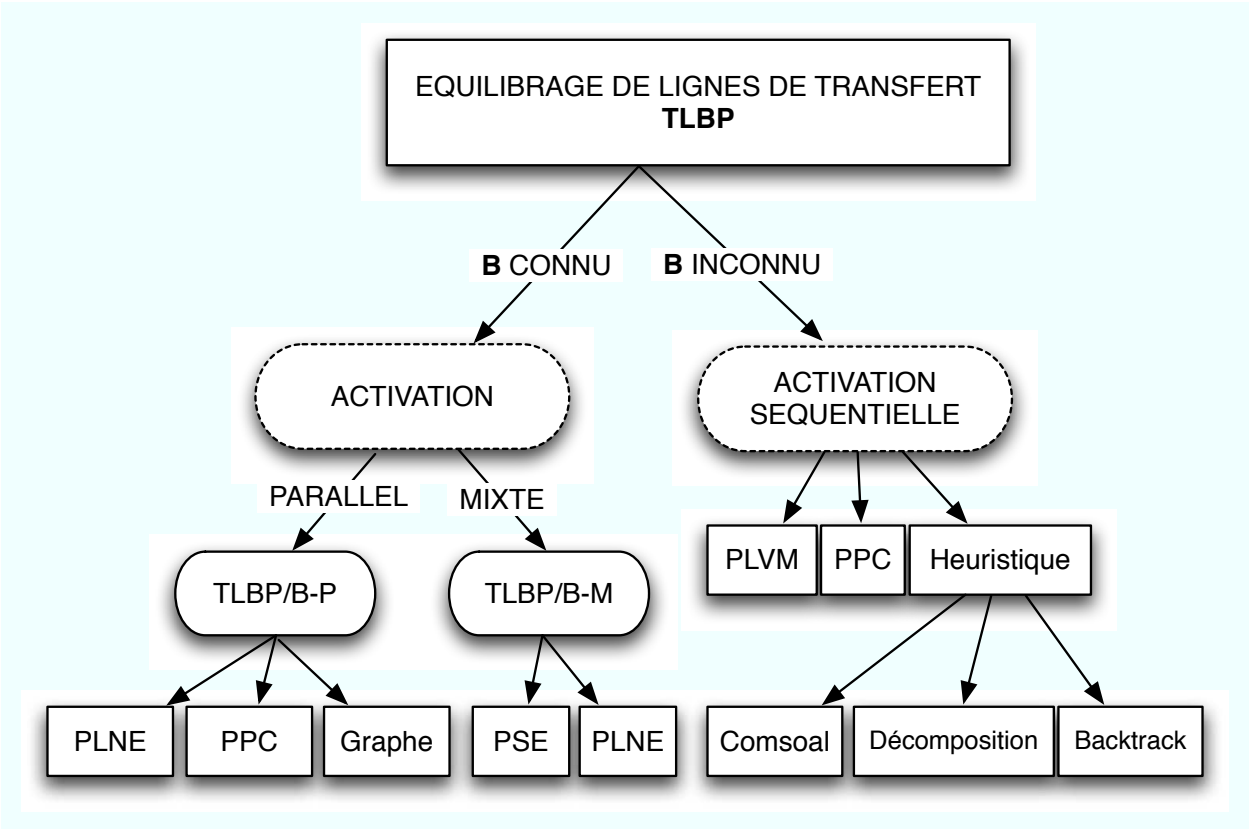


FIG. 3.1 – Classification des problèmes de type TLBP

Le schéma de la figure 3.1 récapitule et classifie l'ensemble des travaux pour les différentes

variantes de TLBP en distinguant ceux qui ont été proposés dans le cadre de cette thèse par un double cadrage.

Dans la suite, nous présentons des travaux dans notre équipe en attirant l'attention sur le fait que ces travaux ont été réalisées en parallèle aux travaux de cette thèse. L'objectif de cette démarche était de proposer un large panel de méthodes pour la résolution du problème traité afin de comparer et d'évaluer les points forts ainsi que les faiblesses de chacune d'entre elles. L'existence d'autres approches pour le problème traité permettent également de valider les résultats obtenus dans cette thèse.

3.6.1 Une approche graphe pour le TLBP/B-P

Dans [DGL06], les auteurs proposent une approche basée sur la programmation dynamique (graphe) pour résoudre le TLBP/B-P. Nous avons consacré la majeure partie de cette thèse à l'étude du même problème, aussi nous reviendrons plus en détail sur sa description (voir section 4.1).

Le TLBP/B-P se situe dans la catégorie des problèmes pour lesquels l'ensemble des têtes (unités) d'usinage sont connues à l'avance (voir classification 3.1). Rappelons, qu'il s'agit de sélectionner un sous-ensemble à partir de l'ensemble de départ de manière à exécuter les opérations nécessaires à la fabrication du produit en respectant toutes les contraintes. Plusieurs types de contraintes sont prises en compte, certaines sont propres aux opérations telles que les précédences et les inclusions, d'autres sont relatives aux unités d'usinage telles que les incompatibilités et les contraintes sur le nombre maximum d'unités par station. De plus, un nombre maximal de stations à ne pas dépasser sur la ligne est également imposé. L'objectif est de minimiser le coût total de la ligne qui est composé du coût des unités et du coût des stations.

L'algorithme suggéré transforme le TLBP/B-P en un problème de recherche du plus court chemin sous contraintes dans un graphe orienté. Les sommets de ce graphe correspondent à des solutions partielles, chacune d'elles caractérise un état représentant l'ensemble des opérations déjà affectées.

La progression dans la construction du graphe se fait en ajoutant un arc sortant d'un sommet existant vers un nouveau sommet s'il existe un chargement possible d'une nouvelle station par rapport à l'état du sommet existant. En d'autres termes, pour chaque nœud développé, une nouvelle station est ouverte, celle-ci est chargée complètement puis elle est fermée. Ce chargement doit respecter l'ensemble des contraintes du problème à l'exception du nombre maximum de stations. L'arc ainsi ajouté est valué avec le coût correspondant à cette affectation comprenant le coût d'une station additionnelle ainsi que le coût des unités qui y ont été affectées. Ce processus est réitéré jusqu'à ce que toutes les possibilités de chargements, pour chacune des stations, aient été explorées.

Le graphe G ainsi construit contient un ensemble de chemins X, chacun d'eux représentant une solution complète allant du sommet de départ au sommet final.

Le problème initial est ramené au problème de recherche du chemin le plus court dans le

graphe G avec au plus m_0 arcs où m_0 est le nombre maximum de stations sur la ligne. Le problème est formellement décrit comme suit :

$$Minimiser \sum_{k=1}^{m(x)} C(d_k(x)) \tag{3.8}$$

$$x \in \mathbf{X} \tag{3.9}$$

$$m(x) \leq m_0 \tag{3.10}$$

où $d_k(x)$ est l'arc k du chemin x, $C(d_k(x))$ est le coût associé à l'arc $d_k(x)$ et $m(x)$ est le nombre de stations pour x. La construction du graphe a été améliorée en intégrant plusieurs règles de dominance.

3.6.2 Un algorithme de type PSE pour le TLBP/B-M

Cette approche a été proposée dans [DI05], elle est désignée par PSE dans le schéma des travaux pour le TLBP (voir figure 3.1). Le problème étudié est le TLBP/B-M, rappelons que dans cette variante l'ensemble des unités d'usinage est également supposé connu au préalable. Quant à l'activation des unités d'une station, elle est mixte, pour plus de détail se référer à la section 1.5.2.

L'algorithme global est une PSE qui s'appuie sur une politique de branchements orienté blocs. Un branchement correspond ainsi à l'affectation d'une unité à la dernière station ouverte. Le choix du prochain nœud à brancher est de type *Best Lower Bound* de façon à développer d'abord le nœud ayant la borne inférieure minimale.

La borne inférieure sur l'objectif est intégrée à chaque nœud de l'arbre. Pour son obtention, il faut détenir d'une part une borne inférieure sur le nombre de stations et d'autre part une borne sur le coût des unités d'usinage qui restent à affecter.

Le calcul de la borne inférieure sur le nombre de stations s'appuie sur l'exploitation des contraintes d'exclusion, celles des précédences, le nombre maximum d'unités par station et temps de cycle maximum. L'idée est de déduire à partir de ces contraintes des ensembles de blocs qui ne peuvent partager une même station. Plus précisément, les graphes de précédence et d'exclusion sont agrégés dans un graphe commun après une transformation du graphe de précédence. Le complément du graphe est ainsi obtenu, les composantes sont identifiées et une borne sur le nombre de stations pour chacune des composantes est estimée. La borne sur le nombre total de stations est la somme des bornes de chacune des stations.

Pour la borne sur le coût des unités, un problème de set partitioning est défini pour déterminer l'ensemble optimal des unités d'usinage à sélectionner parmi celles qui restent. Plus exactement, le problème de set partitioning se pose comme la détermination d'un sous-ensemble d'unités à partir de celles non encore affectées pour couvrir l'ensemble des opérations résiduelles (les opérations absentes de la solution partielle correspondante). Le sous-ensemble d'unités ayant le coût minimum est obtenu en résolvant un problème de set partitioning en appliquant une méthode PSE. Le coût optimal du set partitioning est une borne inférieure pour le coût des unités du problème initial le TLBP/B-M.

3.7 Conclusion

Dans ce chapitre, nous avons repris les travaux relatifs à l'équilibrage des lignes d'assemblage, en particulier ceux concernant le SALBP, en raison des points de similitudes qu'ils présentent avec les problèmes traités dans ce mémoire. En particulier, il s'agit, dans les deux cas, de trouver la meilleure affectation des opérations aux postes de travail. Il reste néanmoins des différences importantes qui font que les méthodes proposées pour l'équilibrage des lignes d'assemblage ne sont plus valides pour le problème de configuration des lignes d'usinage qui nous intéresse. Nous rapportons dans les points suivants, les divergences essentielles avec le SALBP :

1. les regroupements d'opérations en blocs ne sont pas considérés dans le SALBP,
2. il n'y a pas de choix possible pour effectuer une opération donnée,
3. l'ordre d'exécution des opérations est toujours séquentiel au sein d'une station,
4. l'objectif du SALBP est de minimiser le nombre de stations, car dans l'assemblage il reflète le coût de la ligne en raison de la présence d'opérateurs humains (salaires), tandis que pour les lignes d'usinage il faut tenir compte, en plus, du coût de l'équipement.

Concernant les travaux de [BT00] pour le choix d'équipement, au delà des différences (1) et (3), citées pour le SALBP, il subsiste le fait de ne considérer qu'un seul type d'équipement par station.

Nous avons ensuite repris les problèmes proches qui ont été traités dans le cadre de l'équilibrage des lignes d'usinage en mettant l'accent sur les travaux effectués au sein de notre équipe. Nous avons suggéré une classification pour positionner l'ensemble des problèmes étudiés. Puis, nous avons abordé les travaux de [Fin04] en montrant les différences essentielles qui font que les modèles proposés ne sont plus applicable pour le TLBP/B-P étudié dans ce mémoire. En particulier, nous soulignons le fait que dans le TLBP il faut déterminer les blocs d'opérations pendant le processus d'optimisation. En effet, il s'agit de trouver les blocs d'opérations pour lesquels il faut construire des unités d'usinage par la suite ce qui situe le TLBP à une phase antérieure à celle qui nous intéresse lors de la résolution du TLBP/B-P

Enfin, nous avons décrit les travaux qui concernent les mêmes problèmes que ceux qui nous intéressent dans la présente thèse, à savoir : le TLBP/B-P et TLBP/B-M. Ces méthodes nous ont aidé à valider respectivement nos modèles et en évaluer les performances.

Deuxième partie

Le TLBP/B-P

Chapitre 4

Définition du problème

Dans le présent chapitre, nous décrivons le problème d'équilibrage de lignes d'usinage de type TLBP/B-P (voir figure page 60). Nous rappelons que ces lignes sont composées de stations dont les unités d'usinage sont activées de façon parallèle. Nous expliquons en détails les données ainsi que les différentes contraintes qui caractérisent ce problème en indiquant l'ensemble des notations utilisées. Pour illustrer l'ensemble de ces notions nous proposons un exemple concret qui nous provient de notre partenaire industriel.

4.1 Définition du problème

Le TLBP/B-P est relatif à l'étude de lignes mono-produit considérant un ensemble fixe d'opérations à effectuer en respectant l'ensemble des contraintes. Il est important de signaler que l'unicité du type de produit considéré est conceptuelle, cette hypothèse n'exclut donc pas la considération physique de plusieurs types de pièces à condition que l'ensemble se comporte comme un seul produit. Plus précisément, si un ensemble fixe d'opérations est à effectuer sur la ligne de manière cyclique en respectant un ensemble de contraintes alors le problème peut être classé comme mono-produit. Par exemple, l'utilisation de palette contenant plusieurs types de pièces illustre bien la notion de mono-produit conceptuel. Dans ce cas, la ligne fabrique plusieurs types de pièces qui sont fixées sur une palette servant de support. La palette peut être alors considérée comme un seul produit car il s'agit d'effectuer les mêmes opérations sur chaque palette.

Dans ce problème, le mode d'activation est en parallèle pour les unités d'usinage sur chacune des stations de manière à ce que le début d'un cycle est désigné par l'enclenchement simultanée de toutes les unités de la ligne et la fin du cycle est marqué par leur terminaison.

Pour résoudre le TLBP/B-P, il faut déterminer la structure de la ligne en termes d'unités d'usinage à mettre en place, sur chaque station, et de nombre de stations à ouvrir en tenant compte de plusieurs contraintes. Une des contraintes est la cadence de la ligne, celle-ci est déterminée par le temps de cycle de la ligne. En pratique, une valeur maximale sur le temps de cycle de la ligne est obtenue à partir de la productivité visée. Cette valeur a pour objectif

de poser une limite sur le temps de travail pour chacune des stations (celui de la station goulot déterminant le temps de cycle effectif). Ainsi, le temps de cycle effectif ne doit jamais dépasser ce seuil pour pouvoir obtenir la productivité visée.

Par ailleurs, plusieurs autres types de contraintes sont définies, nous distinguons celles qui concernent directement les opérations, par exemple, les contraintes de précédence, de celles qui sont relatives aux unités d'usinage telles que les contraintes d'incompatibilité (exclusion).

Le problème d'optimisation se pose en termes de sélection d'un sous-ensemble d'unités d'usinage, à partir de l'ensemble d'unités disponibles, de telle façon que toutes les opérations soient effectuées, que toutes les contraintes soient respectées et que le coût total soit minimal. Ce coût correspond à la somme des coûts engendrés par la mise en place des stations et le coût des unités d'usinage choisies.

Concrètement, le problème d'optimisation doit apporter des réponses aux interrogations suivantes :

1. Considérant les coûts des unités d'usinage et des stations de travail, quel est le coût minimum à investir dans l'équipement ?
2. Quelle est la configuration associée à ce coût optimal, c'est-à -dire :
   (a) Combien de stations de travail faut-il mettre en place ?
   (b) Combien et quelles sont les unités d'usinage qui doivent être utilisées ?

Pour ce problème, nous développons une approche basée sur la programmation par contraintes (PPC) et deux programmes linéaires en nombres entiers (PLNE) l'un est orienté blocs et l'autre est orienté opérations. Ci-dessous, nous présentons les notations des données que nous utiliserons pour la formulation de ces modèles. Nous introduisons également les ensembles, les collections ainsi que les graphes que nous employons pour représenter les différentes contraintes du problème.

4.1.1 Les données

Les données du problème sont constituées de l'ensemble des opérations à effectuer sur la ligne, de l'ensemble des unités d'usinages disponibles, des coûts et des temps d'exécution des unités d'usinage et du coût d'ouverture d'une station.

L'ensemble des opérations

Toutes les opérations qui doivent être effectuées sur la ligne sont connues. Nous les désignons par l'ensemble $\mathbf{N} = \{1, \ldots, n\}$.

L'ensemble des unités d'usinage (blocs)

Nous employons le terme *bloc* d'opérations pour simplifier le vocabulaire, aussi lorsque nous faisons référence à l'appartenance d'une opération à un bloc nous entendons que l'unité d'usinage correspondante exécute l'opération en question.

L'ensemble des blocs disponibles **B** est défini comme suit :

$\forall b \in \mathbf{B}$, tel qu'il existe une unité d'usinage effectuant toutes les opérations de b, notées $\mathcal{N}(b)$ tel que $\mathcal{N}(b) \subseteq \mathbf{N}$.

Ainsi, chaque bloc b est défini par l'ensemble d'opérations $\mathcal{N}(b)$ qu'il contient.

Dans le cas d'une conception préliminaire, cet ensemble est construit suite à une étude de disponibilité sur le marché ou à une analyse préalable de faisabilité des têtes d'usinage, tandis que pour la reconfiguration d'une ligne une partie de **B** correspondra aux blocs qui sont déjà mis en fonctionnement sur la ligne. Selon les besoins nécessitant la reconfiguration, cet ensemble est éventuellement complété par des blocs qui effectueront des opérations supplémentaires, si de telles opérations doivent être introduites après la reconfiguration. De la même façon, si la ligne, suite à la reconfiguration, ne doit plus effectuer certaines opérations, alors les unités correspondant à ces opérations sont à supprimer de l'ensemble des disponibilités **B**. L'ensemble **B** peut toutefois être mis à jour de façon à intégrer de nouvelles têtes d'usinages qui sont apparues sur le marché ou qui sont conçues par le bureau d'études de l'entreprise elle-même.

En général dans **B**, il existe plusieurs blocs possibles pour chaque opération, aussi une décision du problème revient à déterminer quel est le bloc à retenir pour chaque opération i de **N**.

Pour le TLBP/B-P, la sélection d'un bloc entraîne l'exécution de l'intégralité des opérations appartenant à ce bloc. Autrement dit, il n'est pas possible de choisir un bloc pour exécuter seulement un sous-ensemble de ses opérations. Cette hypothèse n'est pas restrictive vu qu'il est toujours possible de considérer des sous-ensembles d'opérations de l'unité correspondante en tant que des éléments de **B** à part entière.

Coût des blocs

Le coût du bloc b est donné par q_b qui représente, selon le cas, soit le coût d'acquisition de l'unité d'usinage correspondante, soit une pondération à ajuster par le concepteur lui-même. Par exemple, en accordant un coût relativement bas à une unité, le décideur est en mesure de formuler sa préférence pour le choix de cette unité. Cette pondération permet d'intégrer de nombreux paramètres, entre autres, l'amortissement des unités qui sont déjà en fonctionnement sur la ligne ou le coût de leur exploitation. Nous ne nous attarderons pas sur l'ajustement ou le calcul de ces coûts. Il est toutefois à noter que cette question peut faire, à elle seule, l'objet de plusieurs études.

Coût des stations

Le coût pour la mise en place d'une nouvelle station est noté C. Nous supposons que ce coût est le même pour toute station car il s'agit du coût des éléments de base tels que le bâtis. Nous employons indifféremment les termes *établir* ou *ouvrir* pour l'introduction d'une nouvelle station dans la ligne. À chaque ouverture d'une nouvelle station le coût C est ajouté

au coût total. De même, le coût des unités d'usinage qui vont équiper cette station est ajouté au coût total de la ligne.

Temps d'exécution des blocs

Chaque bloc est caractérisé par son coût mais aussi par son temps d'exécution. Ce temps est fourni lors de l'acquisition du bloc et correspond, en pratique, au temps nécessaire pour effectuer toutes les opérations du bloc. Ainsi, pour les unités multi-broches, il peut être estimé, à une constante près[1], au plus grand temps d'exécution de ses opérations [DFG+06b].

4.1.2 Les contraintes

Différentes contraintes sont prises en compte, notamment le temps de cycle qui assure une productivité voulue de la ligne, les contraintes classiques de précédence entre les opérations ainsi que des contraintes d'inclusion pour les opérations et d'exclusion pour les blocs.

Le temps de cycle (productivité)

T_0 représente le temps de cycle maximum ou cadence maximale de la ligne (capacité de production). Du fait que les lignes sont synchrones, le temps de cycle effectif correspond au temps écoulé entre le moment où les stations activent leurs unités et le moment où elles terminent toutes leurs opérations. Autrement dit, le temps de cycle correspond au temps qui sépare la sortie de deux pièces de la ligne. Dans le cas du TLBP/B-P, le temps de cycle est déterminé par l'unité qui a le plus grand temps d'exécution (se référer à la section 1.5.4) auquel il faut ajouter le temps constant qui est nécessaire au mécanisme de transfert pour déplacer une pièce d'une station à une autre. Afin que le temps de cycle soit respecté, il faut que $T \leq T_0$, tel que T est le temps de cycle effectif de la ligne. Pour la cas parallèle cette contrainte peut être traité avant la résolution du problème. En effet, il suffit d'éliminer chacun des blocs ayant un temps d'exécution supérieur à T_0.

[1] La constante permet de prendre en compte l'avance à vide des têtes d'usinage et le temps de transport.

Contraintes de précédence

D^{or} est l'ensemble réunissant les contraintes de précédence liant certaines opérations de $\mathbf{N}$. Ses éléments sont des couples d'opérations tels que tout couple $(i, j) \in D^{or}$ représente une contrainte d'antériorité entre l'opération i et l'opération j. Plus exactement, l'opération j ne peut commencer son exécution avant que l'opération i ne termine la sienne. Notons que ces contraintes peuvent être représentées par un graphe orienté $G^{or} = (\mathbf{N}, D^{or})$. Dans le cas d'une activation parallèle des unités multi-broche, toutes les opérations effectuées sur une même station sont exécutées simultanément. Ainsi, aucun couple $(i, j) \in D^{or}$ ne peut être assigné à une même station. Plus exactement, l'opération i doit être affectée à une station qui précède celle de l'opération j.

Contraintes d'inclusion

D^{in} est la collection des contraintes d'inclusion entre les opérations où chaque élément est un sous-ensemble $d \subset \mathbf{N}$ d'opérations qui doivent être exécutées sur la même station. Pour représenter graphiquement ces contraintes, nous utilisons le graphe non orienté $G^{oi} = (\mathbf{N}, A_1)$ tel que les nœuds représentent les opérations et un arc de A_1 est construit entre tout couple d'opérations appartenant à un sous-ensemble d'inclusion $d \in D^{in}$. Il est intéressant de noter que tout ensemble d appartenant à D^{in} correspond forcément à une clique dans le graphe G^{oi} et que la relation d'inclusion est transitive.

Contraintes d'exclusion

D^{ex} représente la collection des contraintes d'incompatibilité entre blocs, nous utilisons le terme contraintes d'exclusion pour y faire référence. La collection D^{ex} comporte des sous-ensembles $E \subset \mathbf{B}$ tels que chaque sous-ensemble correspond à un ensemble de têtes qui ne peuvent, en aucun cas, être installées sur une même station. Cependant, l'interdiction porte sur l'affectation de l'intégralité des éléments de E et n'affecte pas ses propres sous-ensembles E' (des sous-ensemble $E' \subset E$ peuvent donc être affectées à la même station, c'est-à-dire E est un ensemble minimal). Par exemple, si : $D^{ex} = E = \{b_1, b_2, b_5\}$ alors $\forall E' \subset E$, E' peut être affecté sur une même station si les autres contraintes ne sont pas violées. Ainsi, $\{b_1, b_2\}$, $\{b_1, b_5\}$ ou $\{b_2, b_5\}$ peuvent être assignés dans une même station. Une condition nécessaire pour pouvoir affecter un ensemble quelconque de blocs E sur la même station est que E ne contienne aucun sous-ensemble appartenant à la collection D^{ex}, c'est-à-dire $\nexists E' \subset E$ tel que $E' \in D^{ex}$.

Ces contraintes peuvent être représentées par un graphe $G^{ex} = (\mathbf{B}, A_2)$ où un arc appartenant à A_2 relie deux blocs incompatibles. Il est également à noter que la relation d'incompatibilité n'est pas transitive, ainsi lorsque un bloc b est incompatible avec un bloc b' qui lui-même n'est pas compatible avec b'', dans ce cas b et b'' ne sont pas forcément incompatibles à moins que le couple (b, b'') appartienne à D^{ex}.

Limitation sur le nombre de stations

m_0 est le nombre maximum de stations que peut comporter la ligne.

Limitation sur le nombre de blocs par station

n_0 est le nombre maximum d'unités d'usinage qui peuvent être installées sur une station, ce qui correspond souvent au nombre d'emplacements d'axes possibles.

4.1.3 Solutions réalisables

Une solution du problème correspond à une configuration réalisable, si et seulement si les conditions suivantes sont remplies :

- Chacune des opérations de **N** doit être exécutée exactement une seule fois. C'est-à-dire qu'aucune opération ne doit être omise ni dupliquée. Donc, aucune intersection entre les blocs d'opérations n'est autorisée car une solution avec deux blocs non disjoints correspond à une configuration avec des unités d'usinage effectuant les mêmes opérations.
- Le temps de travail de chaque station ne doit pas dépasser le temps de cycle maximum T_0.
- Toutes les autres contraintes sont également respectées : inclusion, exclusion, précédence et limitations sur le nombre de stations et sur le nombre d'unités d'usinage par station.

Parmi toutes les configurations réalisables celle dont la somme des coûts des unités d'usinage retenues pour équiper les stations et des coûts engendrés par la mise en place des stations est minimale est la solution optimale. Il est cependant courant d'avoir plusieurs solutions optimales, celles-ci ont le même coût mais pas forcément la même structure. Ces structures peuvent être différentes car elles ne contiennent pas les mêmes blocs ou parce que l'affectation des blocs aux stations est différente.

4.2 Un exemple numérique

Afin d'illustrer toutes les données et les contraintes décrites précédemment, nous présentons un exemple industriel concret qui s'est posé lors de la conception d'une ligne d'usinage pour la fabrication de la pièce décrite dans la figure 4.1.

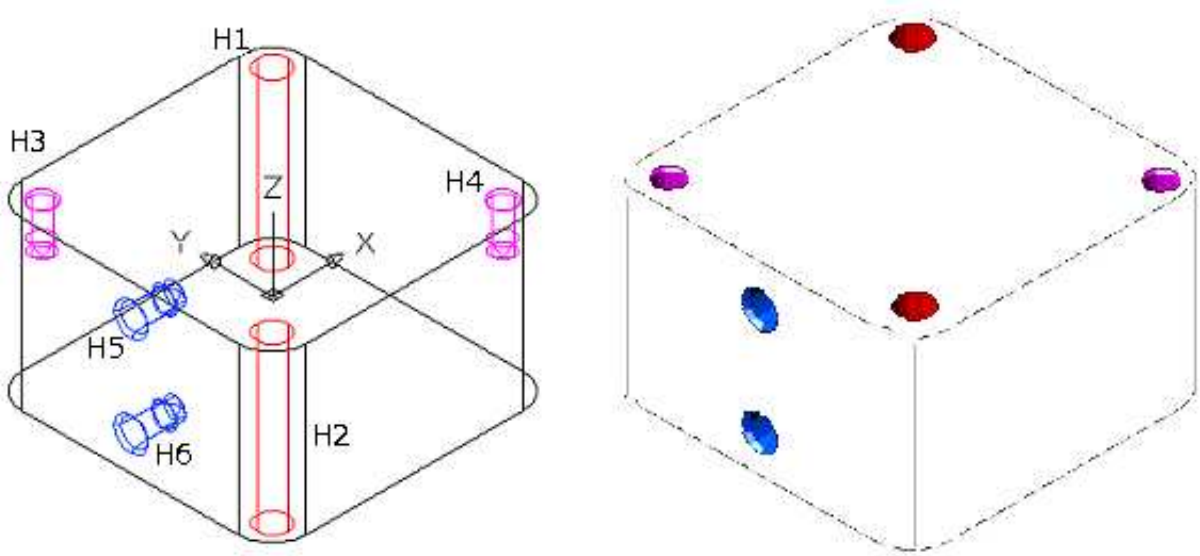


FIG. 4.1 – Deux vues de la pièce

Pour des raisons de confidentialité, les coûts rapportés ont été modifiés, mais toutes les autres données et contraintes sont authentiques.

Pour la fabrication de la pièce, 14 opérations sont indispensables, elles sont décrites dans le tableau 4.1. Pour ce faire, on dispose de 21 têtes d'usinage qui sont décrites dans le tableau 4.2. Le coût pour mettre en place une station est 15500 unités monétaires. De plus, le nombre maximum de blocs par stations $n_0 = 2$ et le nombre maximum de stations sur la ligne $m_0 = 8$.

opération	description	paramètres	temps opératoire
1	Perçage de H1	∅ 14	169,60
2	Perçage de H2	∅ 14	169,60
3	Perçage de H3	∅ 9,8	48,60
4	Chambrage du chanfrein H3	∅ 12	7,30
5	Alésage H3	∅ 10	13,71
6	Perçage de H4	∅ 9,8	48,60
7	Chambrage du chanfrein H4	∅ 12	7,30
8	Alésage H4	∅ 10	13,71
9	Perçage H5	∅ 10,2	52,42
10	Chambrage du chanfrein H5	∅ 13,2	8,04
11	Filetage H5	M12-6G	3,21
12	Perçage H6	∅ 10,2	52,42
13	Chambrage du chanfrein H6	∅ 13,2	8,04
14	Filetage H6	M12-6G	3,21

TAB. 4.1 – L'ensemble **N** des opérations

blocs	opérations	coût	temps	blocs	opérations	coût	temps
b_1	1,3,6	12200	169,6				
b_2	2,4,7	10100	169,6	b_{12}	1,2,4,7	14150	169,6
b_3	1,2,3,6	14400	169,6	b_{13}	3,6	10000	48,60
b_4	4,7	7950	7,30	b_{14}	1,6	10200	169,6
b_5	9,12	10200	52,42	b_{15}	1	10900	169,6
b_6	10,13	7450	8,04	b_{16}	2	10900	169,6
b_7	5,8	9000	13,71	b_{17}	1,2	13800	169,6
b_8	11	9750	3,21	b_{18}	3	9000	48,60
b_9	14	9750	3,21	b_{19}	6	9000	48,60
b_{10}	2,3,6	12300	169,6	b_{20}	5	8500	13,71
b_{11}	1,4,7	10150	169,6	b_{21}	8	8500	13,71

TAB. 4.2 – L'ensemble **B** des unités d'usinage

L'ensemble des **contraintes de précédence** sont données comme suit :

$D^{or} = \{(3,4),(4,5),(6,7),(7,8),(9,10),(1,11),(2,11),(5,11),(8,11),(10,11),(13,11),$
$(12,13),(1,14),(2,14),(5,14),(8,14),(10,14),(13,14)\}$.

La figure 4.2 montre le graphe $G^{or} = (\mathbf{N}, D^{or})$ représentant les contraintes de précédence.

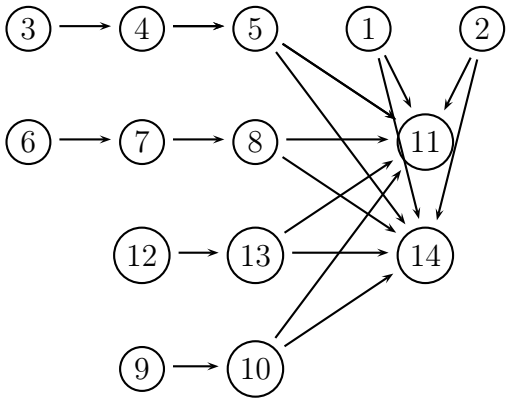


FIG. 4.2 – Le graphe de précédence G^{or}

Les **contraintes d'inclusion** sont décrites comme suit :

$D^{in} = \big\{\{1,2\},\{3,6\}\big\}$.

Le graphe G^{in} permet de représenter les contraintes d'inclusion comme indiquée dans la figure 4.3.

L'ensemble des **contraintes d'exclusion** est donné par la collection D^{ex} qui ne contient que des couples de blocs incompatibles. Toutefois, le modèle prend en compte le cas général où plus de deux blocs peuvent être incompatibles (le graphe correspondant est donné par

4.4).

FIG. 4.3 – Le graphe d'inclusion G^{in}

$D^{ex} = \big\{\{5,8\},\{6,8\},\{9,8\},\{1,7\},\{2,7\},\{3,7\},\ \{4,7\},\{5,7\},\{6,7\},\{10,7\},\{11,7\},$
$\{13,7\},\{14,7\}\ ,\{15,7\},\{16,7\},\{17,7\},\ \{18,7\},\{19,7\},\{20,1\},\{20,2\},\{20,3\},\{20,4\},$
$\{20,6\},\{20,10\},\{20,11\},\{20,12\}\ \{20,13\},\{20,14\},\{20,15\},\{20,16\},\{20,17\},$
$\{20,18\},\{20,19\},\{21,1\},\{21,2\},\{21,3\},\{21,4\},\{21,5\},\{21,6\},\{21,10\},\{21,11\},$
$\{21,12\},\{21,13\},\{21,14\},\{21,15\},\{21,16\},\{21,17\},\{21,18\},\{21,19\},\{20,5\}\big\}$

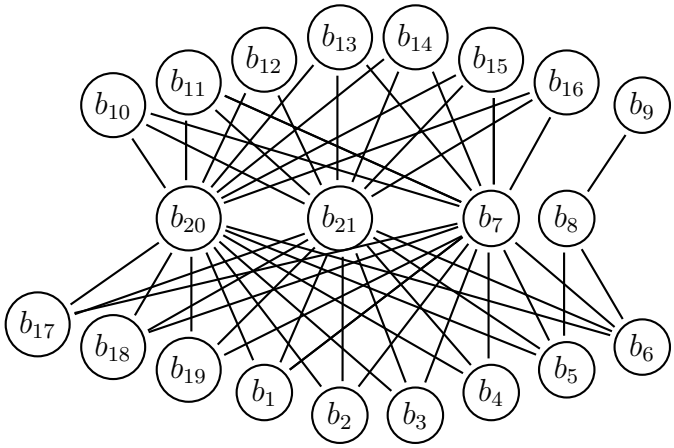


FIG. 4.4 – Le graphe d'exclusion G^{ex}

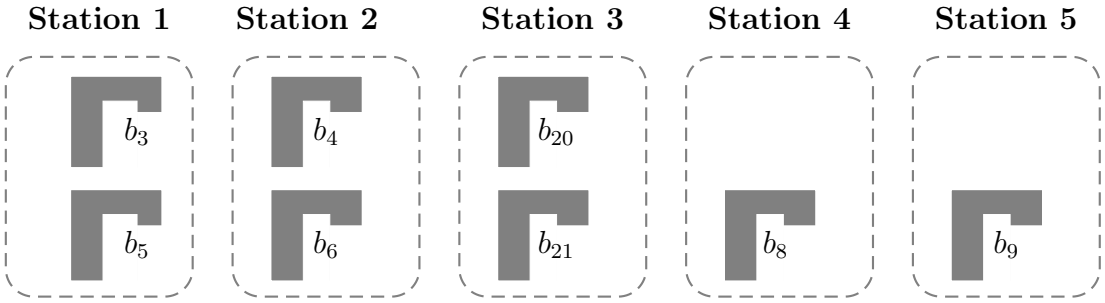


FIG. 4.5 – Une affectation réalisable des blocs aux stations

Une solution réalisable est donnée dans la figure 4.5 ayant un temps de cycle de 169,6. Elle est composée de 5 stations : la première est équipée des deux blocs b_3 et b_5, la seconde contient b_4 et b_6, la troisième est composée de b_{20} et b_{21}, la quatrième comporte b_8 et la dernière b_9. Par conséquent, le coût total est de 154000. Il est intéressant de constater que les quatre premières stations sont établies en raison des contraintes de précédence. Par exemple, le bloc b_{20} et b_8 ne peuvent pas partager la même station car b_8 contient une opération successeur de b_{20}. De même b_9 est successeur de b_{20} et b_{21}. Une dernière station doit être ouverte pour contenir b_9 car d'une part b_9 est successeur de b_{20} et b_{21}et d'autre part b_9 ne peut partager une station avec b_8 en raison de l'existence d'une incompatibilité qui les lie (voir l'arc b_8,b_9 dans le graphe G^{ex}).

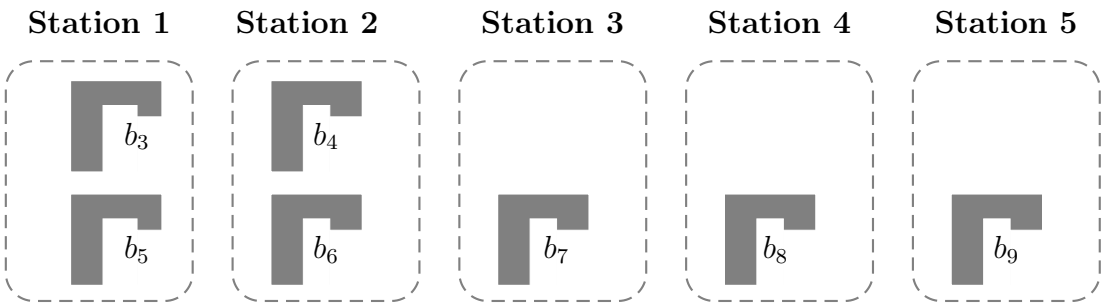


FIG. 4.6 – La meilleure affectation des blocs aux stations

La solution optimale est montrée dans la figure 4.6 où le choix du bloc b_7 se substitue aux blocs b_{20} et b_{21} diminuant le coût total à 146000.

Il est important de souligner que pour respecter le temps de cycle dans le cas du TLBP-B/P il suffit de filter l'ensemble **B** en écartant les blocs ayant un temps d'exécution strictement supérieur au temps de cycle maximum.

4.3 Pré-traitements

Dans cette section, nous proposons deux types de pré-traitements pour le modèle présenté ci-dessus, en premier lieu nous présentons une analyse de la consistance des données qui va permettre d'éliminer certains éléments de l'ensemble **B** et des contraintes d'inclusion et d'exclusion. Par la suite, nous proposons d'effectuer quelques tests (avant d'entamer le processus de résolution) afin de détecter des infaisabilités, évitant ainsi de lancer une recherche assurément infructueuse [BDGL05a, BDGL06].

4.3.1 Consistance

L'étude de la consistance des données a pour objectif de réduire l'espace de recherche en soustrayant les éléments qui engendrent des tentatives infructueuses. Plus précisément, nous appliquons des règles d'inconsistance pour éviter de faire participer des éléments provoquant systématiquement une violation d'une des contraintes. Cette étude sert à diminuer l'effort de recherche en gardant indemne l'ensemble des solutions réalisables et en éliminant le maximum d'éléments inconsistants. Nous proposons d'appliquer des procédures basées sur l'étude de la consistance des données d'une part et sur la dominance de certaines contraintes d'autre part. Afin de modérer les efforts, nous préférons employer ces techniques en tant que pré-traitements, c'est-à-dire avant même d'entamer la résolution du problème au lieu de les effectuer à chaque nœud de l'arbre de résolution.

Nous décrivons d'abord les pré-traitements que nous proposons pour réduire l'ensemble des blocs disponibles $\mathbf{B}$ en expliquant, dans chaque cas, les raisons qui ont permis de conclure sur l'inconsistance des blocs à écarter. Ensuite, nous appliquons quelques règles pour vérifier la faisabilité du problème.

Inconsistance des blocs

1. Si le temps d'exécution du bloc b est strictement supérieur au temps de cycle T_0 alors ce bloc est inconsistant et peut être supprimé de l'ensemble des blocs disponibles $\mathbf{B}$.
   En effet, comme l'activation des blocs se fait en parallèle, il en résulte que le temps de cycle est déterminé par le bloc qui a le plus grand temps d'exécution. Par conséquent, tout bloc avec un temps supérieur à T_0 n'assurerait pas la productivité minimale requise et sa sélection conduirait à la violation du temps de cycle. Ceci est une condition suffisante pour éliminer ce bloc afin d'assurer la consistance de $\mathbf{B}$.
   Il en résulte que les contraintes d'exclusion qui concernent ce bloc deviennent caduques car elles font intervenir un bloc non existant, elles sont donc à supprimer. Toutefois, les contraintes qui font intervenir les opérations effectuées par ce bloc doivent, quant à elles, être maintenues[2], car ces contraintes sont propres aux opérations qui peuvent appartenir, par ailleurs, à d'autres blocs[3]. Ainsi, on écrit :

   $\forall b \in \mathbf{B}$,
   Si
   $t_b > T_0$
   alors
   $\mathbf{B} := \mathbf{B} - \{b\}$
2. Dés lors les opérations d'un même bloc sont effectuées de façon simultanée par la même tête d'usinage multi-broche tout bloc contenant des opérations reliées par une

[2]Le problème engendré, dans le cas de la suppression des contraintes liées aux opérations du bloc b, serait une relaxation du problème original car des contraintes intrinsèques aux opérations auraient été omises. En terme d'espace de solutions, les problèmes sont différents.

[3]Si le problème admet une solution réalisable, il est forcé que ces opérations appartiennent à d'autres blocs.

relation de précédence peut être supprimé de **B**. En effet, si un tel bloc est considéré la contrainte d'antériorité ne pourra être respectée. Plus formellement, on écrit :

$\forall b \in \mathbf{B}$,
Si
$\exists i, j \in \mathcal{N}(b)$ tel que : $(i, j) \in D^{or}$
alors
$\mathbf{B} := \mathbf{B} - \{b\}$

3. Si un bloc b contient au moins une opération en commun avec un ensemble $d \in D^{in}$ alors la sélection de ce bloc implique l'exécution des opérations de $\mathcal{N}(b) \bigcup d$ dans la même station. En effet, les opérations de $d \in D^{in}$ sont indissociables et les opérations appartenant à un même bloc le sont également. Si l'intersection de $\mathcal{N}(b)$ et d est non vide alors c'est l'intégralité de l'union qui devient indissociable et doit, donc, être exécutée sur la même station. Par conséquent, le choix de b ne peut amener à une solution faisable lorsqu'une relation de précédence existe entre certaines opérations de cette union. Il en résulte, que tout bloc ayant une intersection avec un $d \in D^{in}$ et contenant une opération précédant ou succédant à une opération dans d est inconsistant. Plus formellement :

$\forall b \in \mathbf{B}, \forall d \in D^{in}$,
Si
$\exists i \in \mathcal{N}(b), \exists j \in d \mid \mathcal{N}(b) \cap d \neq \varnothing \wedge ((i, j) \in D^{or} \vee (j, i) \in D^{or})$
alors
$\mathbf{B} := \mathbf{B} - \{b\}$

4. Lorsque un bloc b contient au moins deux opérations et que ces dernières sont liées par des contraintes de précédence aux opérations d'un sous-ensemble d'inclusion ; dans ce cas, si le sens des deux relations d'antériorité est contradictoire alors, quelle que soit l'affectation du bloc b, les deux contraintes ne pourraient être satisfaites simultanément. Plus formellement, ce cas s'écrit de la façon suivante :

$\forall b \in \mathbf{B}, \forall d \in D^{in}$
Si
$\exists k, l \in d, \exists i, j \in \mathcal{N}(b) \mid (i, k) \in D^{or} \wedge (l, j) \in D^{or}$
alors
$\mathbf{B} := \mathbf{B} - \{b\}$

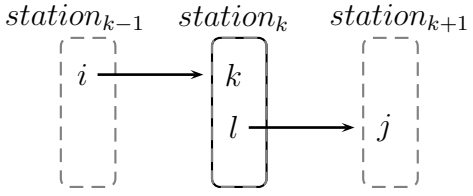


FIG. 4.7 – Inconsistance des blocs contenant i et j

En effet, dans ce cas, les opérations i et j doivent être exécutées sur deux stations distinctes et séparées par au moins une autre station, car les opérations k et l sont indissociables vu qu'elles appartiennent au même sous-ensemble d'inclusion. En raison des contraintes de précédence (i, k) et (l, j), le bloc b ne peut appartenir à une solution sans enfreindre une de ces contraintes. Le schéma donné dans la figure 4.7 montre clairement qu'il faut au minimum une station entre celle où l'opération i est affectée et celle de j pour respecter les contraintes de précédence.

5. Lorsqu'il n'existe aucun sous-ensemble $B' \subset \mathbf{B}$ contenant le bloc b tel que toutes les conditions suivantes soient vérifiées :
   a) tous les blocs de B' sont mutuellement disjoints ;
   b) l'union des opérations des blocs de B' coincide avec l'ensemble $\mathbf{N}$;
   c) la cardinalité de B' ne dépasse pas le nombre maximale des blocs qui peuvent être assignés sur la ligne : $m_0 n_0$.

   Plus précisément, le bloc b est éliminé, si $b \in B'$ et :

   $\exists i \in \mathbf{N} - \{\mathcal{N}(b)\}$ tel que $\forall b' \in \mathbf{B} - B', i \in \mathcal{N}(b') \Rightarrow \mathcal{N}(b) \cap \mathcal{N}(b') \neq \varnothing$

   En d'autres termes, si le bloc b est choisi l'opération i ne pourra pas être effectuée car tout bloc b' contenant l'opération i est mutuellement exclusif avec le bloc b car ils ont des opérations communes. Par conséquent, le bloc b est inconsistant et peut être supprimé.

Dans le premiers cas c'est le temps d'exécution du bloc qui le rend inconsistant. Dans les trois cas suivants, nous supprimons des blocs en raison des violations des relations d'antécédence. Quant au dernier cas, il concerne les blocs dont le choix engendre l'absence d'une ou de plusieurs opérations. Dans ce qui suit, nous présentons d'autres règles de pré-traitement.

Dominance de contraintes

Une analyse de dominance de certaines contraintes par rapport à d'autres permet de détecter des contraintes moins fortes ou redondantes. Ainsi, les contraintes dominantes sont conservées alors que les contraintes dominées peuvent être omises car elles sont en quelque sorte contenues dans les premières.

Par exemple, un sous-ensemble E peut être supprimé de l'ensemble D^{ex} lorsque sa cardinalité est supérieure au nombre maximum de blocs par station, *i.e.* $|E| > n_0$. En effet, le nombre de blocs qui peuvent être affectés à la même station étant limité à n_0, il n'est plus nécessaire de conserver l'interdiction exprimée par E car cette contrainte devient redondante.

Par ailleurs, le pré-traitement concernant la collection D^{ex} peut être affiné davantage en supprimant les sous-ensembles $E \in D^{ex}$ qui correspondent au cas suivant :

$\exists b, b' \in E$ tel que $\mathcal{N}(b) \cap \mathcal{N}(b') \neq \varnothing$.

Dans ce cas, du fait que les blocs b et b' ont des opérations communes, ils deviennent mutuellement exclusifs. Ainsi, les contraintes interdisant leur affectation à la même station

deviennent superflues car elles seront déjà prises en compte par les inéquations interdisant l'affectation de plus d'un bloc parmi ceux qui exécutent les mêmes opérations (voir (5.19)).

Réduction de la collection d'inclusion D^{in}

Les sous-ensembles de D^{in} dont l'intersection est non vide peuvent être fusionnés en un seul sous-ensemble. Par définition, un sous-ensemble de D^{in} est un sous-ensemble contenant les opérations de $\mathbf{N}$ qui doivent impérativement être exécutées sur la même station. La relation d'inclusion est transitive c'est-à-dire s'il existe des opérations qui doivent être exécutées avec une certaine opération et que celle-ci elle-même doit être effectuée avec d'autres alors c'est l'union de toutes ces opérations qui devient indissociable.

À présent, nous proposons quelques tests nécessaires mais non suffisants à la faisabilité des instances. Ces tests ont pour objectif de détecter les instances irréalisables avant même d'entamer leur résolution.

4.3.2 Vérification de la faisabilité

Afin qu'une instance du problème admette une solution réalisable il est nécessaire que les conditions suivantes soient vérifiées :

1. Il faut que l'union des blocs de $\mathbf{B}$ coincide avec l'ensemble des opérations, c'est-à-dire il doit y avoir au moins un bloc pour chaque opération, donc :

$$\bigcup_{b \in \mathbf{B}} \mathcal{N}(b) = \mathbf{N}$$

2. Le graphe de précédence G^{or} doit être acyclique. C'est-à-dire que le graphe ne doit pas comporter d'arc liant un sommet j avec un sommet i s'il existe un chemin allant de i vers j.
3. Pour tout sous-ensemble d'inclusion $d \in D^{in}$, il doit exister un sous-ensemble $B' \subseteq \mathbf{B}$ de blocs mutuellement disjoints tel que :
   (a) le nombre de blocs de B' ne dépasse pas le nombre maximum de blocs par station soit : $|B'| \leq n_0$;
   (b) l'intégralité des opérations appartenant à d doit être contenue dans les blocs de B', soit : $d \subseteq \bigcup_{b \in B'} \mathcal{N}(b)$;
   (c) l'ensemble des blocs de B' ne doit pas contenir de blocs incompatibles, c'est-à-dire : $\forall E \subseteq B', E \notin D^{ex}$;
   (d) aucune précédence ne doit lier deux opérations appartenant aux blocs de B', soit : $\forall i, j \in \bigcup_{b \in B'} \mathcal{N}(b), (i, j) \notin D^{or}$.

4.4 Conclusion

Dans le présent chapitre nous avons introduit le problème d'optimisation TLBP/B-P, en précisant les notations ainsi que les données et les contraintes. Nous avons également rapporté un exemple concret, qui nous a été transmis par un de nos partenaires industriels, afin d'illustrer les différentes données et contraintes du problème et de mieux cerner ses caractéristiques.

Nous avons également proposé plusieurs pré-traitements pour réduire la taille des données en éliminant quelques éléments inconsistants. De plus, nous avons suggéré quelques tests qui permettent de détecter à l'avance certaines instances infaisables.

Nous proposons dans le chapitre suivant trois formulations pour modéliser et résoudre ce problème. La première est basée sur la programmation par contraintes et les deux autres sont des programmes linéaires en nombres entiers. Nous proposons également deux bornes inférieures, l'une obtenue par relaxation linéaire d'un des modèles proposés et la seconde par relaxation à un problème particulier de set partitioning.

Chapitre 5

Modélisation et résolution du TLBP/B-P

Dans ce chapitre, nous proposons d'abord une formulation générique. Puis, nous décrivons le schéma global adopté pour une approche basée sur la programmation par contraintes (PPC) en décrivant les règles de propagation qui sont appliquées ainsi que les règles de dominance. Ensuite, deux bornes inférieures sont fournies, l'une est basée sur une relaxation linéaire et la seconde sur une relaxation à un problème particulier de set partitioning. Nous proposons également un modèle basé sur la programmation linéaire en nombres entiers que nous améliorons par la suite en reformulant certaines contraintes. Enfin, nous suggérons des techniques efficaces pour réduire le nombre de variables binaires et augmenter les performances du modèle linéaire.

5.1 Approche basée sur la PPC

Dans cette partie du travail, nous proposons deux bornes inférieures pour le TLBP/B-P : la première est une relaxation à un problème particulier de set partitioning [BDG+04], quant à la seconde, elle est obtenue grâce à une relaxation linéaire d'un des modèles linéaires en nombres entiers que nous proposons dans la suite.

Nous décrivons ensuite le schéma global de l'approche PPC, celui-ci est basée essentiellement sur une méthode de type séparation et évaluation combinée à une règle de dominance. Nous décrivons le modèle utilisé pour cette approche, ensuite nous indiquons la politique de branchement utilisée pour la séparation et l'évaluation.

5.1.1 Une relaxation au problème de set partitioning

Dans cette section, nous proposons une borne inférieure basée sur une relaxation du TLBP/B-P à un problème de set partitioning [BDG+04]. Pour ce faire, nous calculons d'abord une borne inférieure sur le nombre des stations et ensuite une estimation du coût engendré par les blocs sélectionnés.

Nous proposons d'exploiter la structure spécifique du problème pour en déduire une borne

inférieure sur le nombre de stations. Ensuite, nous proposons une relaxation à un problème de partitionnement particulier afin de déduire une borne sur le second terme de l'objectif, à savoir : le coût des blocs sélectionnés. Cette borne est une adaptation de celle proposée dans les travaux de [DI05] où les auteurs se sont intéressés au problème mixte (de type TLBP/B-M, voir la classification dans la figure 3.1).

Nombre de stations

Pour obtenir une borne inférieure sur le nombre de stations nous exploitons d'abord les contraintes d'exclusion entre les blocs avec les contraintes de précédence entre les opérations. Puis, nous renforçons la valeur de la borne en prenant en compte également les contraintes sur le nombre maximum de blocs par station.

Dans un premier temps, nous construisons un graphe non orienté pour les blocs à partir du graphe de précédence G^{or}. Nous notons ce graphe $G^{br} = (\mathbf{B}, E^{br})$. Plus précisément, le graphe G^{br} est obtenu après une transformation de G^{or}, en déduisant les contraintes de précédence entre les blocs à partir de celles qui existent entre les opérations. En particulier, chaque arc qui lie deux opérations est transformé en une ou plusieurs arêtes qui lient les blocs qui les contiennent. Ainsi, s'il existe un arc entre les sommets i et j dans G^{or}, sa correspondance, dans G^{br}, est telle que, tout bloc qui contient l'opération i est lié par une arête à tout bloc qui contient l'opération j. De plus, nous appliquons la fermeture transitive du graphe G^{br} ainsi obtenu[1]. Ainsi, si $\exists(b, b') \in E^{br}$ et $\exists(b', b'') \in E^{br}$ alors nous ajoutons l'arête (b, b'') dans E^{br}.

Ensuite, nous fusionnons les deux graphes G^{ex} et G^{br} pour obtenir le graphe G^B. Ce graphe contient ainsi l'ensemble des contraintes d'exclusion et toutes les précédences reliant les blocs. Puis, nous construisons le complément de ce graphe que nous notons $\overline{G^B}$, tel que $\overline{G^B}$ contient toutes les arêtes qui existent dans le graphe complet[2] et qui n'existe pas dans le graphe G^B. Nous notons $\overline{G^B_k} = (V_k, E^{V_k}), k = 1, \ldots, l$, toute composante connexe du graphe $\overline{G^B}$. Il est clair que deux blocs qui appartiennent à deux composantes distinctes ne peuvent pas être affectés à la même station. Ainsi, le nombre de composantes connexes de $\overline{G^B}$ fournit une borne inférieure sur le nombre de stations. De plus, le nombre de stations nécessaires pour les blocs de chaque composante connexe ne peut pas être inférieure à la cardinalité de l'ensemble indépendant maximum de la composante [Aig95].

$$\omega(\overline{G}^B_k) \geq \sum_{\sigma \in V_k} (1 + deg(\sigma))^{-1} \tag{5.1}$$

où $deg(\sigma)$ est le degré du nœud σ appartenant à la composante $\overline{G}^B_k$.

L'ensemble des blocs de chaque composante connexe $\overline{G}^B_k$ requière donc au moins $\lceil \omega(\overline{G}^B_k) \rceil$ stations.

[1] En effet, le graphe de précédence ne contient pas les arcs redondants, ils sont considérés de façon implicite.

[2] Le graphe complet comporte les mêmes sommets, soit dans ce cas l'ensemble des blocs, et contient toutes les arêtes possibles, en d'autres termes tout couple de sommets est lié par un arc.

Cette valeur peut être renforcée en prenant en compte les contraintes de capacité sur le nombre de blocs par station :

$$\tilde{m}_k \geq \left\lceil \frac{|V_k|}{n_0} \right\rceil \tag{5.2}$$

De (5.1) et (5.2) la borne pour chaque composante $\overline{G}_k^B$ est comme suit :

$$m^* = max \left\{ \lceil \omega(\overline{G}_k^X) \rceil, \left\lceil \frac{|V_k|}{n_0} \right\rceil \right\} \tag{5.3}$$

En additionnant l'ensemble des bornes nécessaires pour chaque composante du graphe, nous obtenons une borne inférieure sur le nombre de stations pour l'ensemble des opérations :

$$\tilde{m} = \sum_{k=1}^{l} \tilde{m}_k \tag{5.4}$$

Coût des blocs

Pour obtenir une borne inférieure sur le coût des blocs, il faut résoudre à l'optimum le problème de set partitioning. Pour ce faire, il est nécessaire de trouver le meilleur sous-ensemble $\beta \subseteq \mathbf{B}$ pour exécuter les opérations non encore affectées, c'est-à-dire $\mathbf{N}' = \mathbf{N} - \{i \mid i \in b, b \in B\}$, tel que $B \subset \mathbf{B}$ est l'ensemble des blocs qui ont déjà été affectés.

Si nous définissons $\mathcal{N}(\beta)$, comme étant l'ensemble des opérations des blocs β, alors $\mathcal{N}(\beta)$ définit l'ensemble des blocs qui restent à affecter, à savoir : $B_2 = \{b \mid b \subseteq N'\}$.

Le sous-ensemble β est alors une partition des opérations non encore affectées en des blocs appartenant à $\mathbf{B} - B$. C'est un problème de set partitioning qui doit vérifier les conditions suivantes :

$$b \in B_2 \tag{5.5}$$

$$\mathcal{N}(b) \bigcap \mathcal{N}(b') = \varnothing, \qquad \forall b, b' \in \beta \tag{5.6}$$

$$\bigcup_{b \in \beta} \mathcal{N}(b) = \mathbf{N}' \tag{5.7}$$

De plus, il faut que la valeur de l'objectif soit minimale :

$$\sum_{b \in \beta} q_b + (m' - 1 + m^*)C \tag{5.8}$$

Nous utilisons une procédure de séparation et d'évaluation pour résoudre le problème (5.5)-(5.8).

5.1.2 Une relaxation linéaire

Cette borne inférieure est obtenue en relâchant les contraintes d'intégrité des variables binaires utilisées dans le modèle linéaire en nombres entiers (5.17)-(5.25) et (5.36)-(5.43) (pour la seconde formulation le coût m^*C induit par l'ouverture sûre de m^* doit être ajouté).

Toutefois, afin que cette borne soit valide dans le schéma global, c'est-à-dire à un nœud donné de l'arbre, il faut intégrer l'ensemble des contraintes qui ont été fixées auparavant. Plus exactement, il faut considérer l'ensemble des blocs qui ont déjà été affectés aux stations et intégrer les contraintes correspondantes dans la relaxation linéaire.

5.1.3 Un modèle générique

Le modèle que nous avons utilisé pour la résolution par PPC est orienté stations. Nous désignons par $L = \{Sta_1, \ldots, Sta_k, \ldots, Sta_m\}$ une solution du problème, tel que : $Sta_k \subset \mathbf{B}$ est l'ensemble des blocs affectés à la station k.

L'objectif qui est exprimé par (5.9) minimise le coût de la lignes composé du coût des m stations ouvertes (le premier terme) ainsi que le coût des blocs sélectionnés (le second terme).

$$Minimiser \quad Cm + \sum_{k=1}^{m} \sum_{b \in Sta_k} q_b \tag{5.9}$$

Pour l'exécution de l'ensemble des opérations nous utilisons la contrainte (5.10). Cette égalité permet de vérifier que l'ensemble constitué de l'union des opérations contenues dans les blocs sélectionnés coincide effectivement avec l'ensemble des opérations à effectuer $\mathbf{N}$.

$$\bigcup_{k=1}^{m} \mathcal{N}(Sta_k) = \mathbf{N} \tag{5.10}$$

Les contraintes (5.11) imposent aux blocs sélectionnés d'être mutuellement disjoints. Ainsi, ces contraintes et les contraintes (5.10) assurent conjointement l'unicité des opérations. De façon plus précise, sans les contraintes (5.11), les solutions pourraient contenir les mêmes opérations plusieurs fois car les contraintes (5.10) imposent uniquement la présence de l'ensemble des opérations sans vérifier leurs duplications éventuelles. Dans ce cas, certaines opérations seraient effectuées plusieurs fois, ce qui est à proscrire.

$$\forall b, b' \in \bigcup_{k=1}^{m} Sta_k, \quad \mathcal{N}(b) \cap \mathcal{N}(b') = \varnothing \tag{5.11}$$

Les contraintes (5.12) imposent le respect des précédences. Nous employons dans ces contraintes la notation $\Gamma^-(b)$ afin de représenter l'ensemble des opérations prédécesseurs de

celles appartenant au bloc b. Ainsi, pour tout bloc b affecté à une station k, nous imposons que ses prédécesseurs soient affectés aux stations antérieures strictement à k.

$$\Gamma^{-}(b) \subseteq \bigcup_{h=1}^{k-1} \mathcal{N}(Sta_h), \quad \forall b \in Sta_k, \forall k = 1, \ldots, m \tag{5.12}$$

Les contraintes d'inclusion pour les opérations sont assurées par (5.13). Elles imposent, plus précisément, à l'ensemble des blocs Sta_k composant une station k de contenir soit toutes les opérations d'un ensemble d'inclusion d, soit aucune d'elles.

$$\Big(\bigcup_{b \in Sta_k} \mathcal{N}(b) \bigcap d \Big) \in \{\varnothing, d\}, \quad \forall d \in D^{in}, \forall k = 1, \ldots, m \tag{5.13}$$

Les contraintes d'exclusion sont exprimées dans (5.14). Elles interdisent aux blocs appartenant à la station Sta_k de contenir un sous-ensemble $E \in D^{ex}$ de blocs incompatibles.

$$E \not\subseteq Sta_k, \quad \forall E \in D^{ex}, \forall k = 1, \ldots, m \tag{5.14}$$

Les contraintes (5.15) interdisent l'affectation de plus de n_0 blocs par station en imposant à chaque ensemble Sta_k d'avoir une cardinalité inférieur ou égale à n_0.

$$|Sta_k| \leq n_0 \tag{5.15}$$

Les contraintes (5.16) vérifient le respect du nombre maximum de stations.

$$m \leq m_0 \tag{5.16}$$

Nous avons implémenté ce modèle à l'aide des libraires de ILOG Solver [ILO].

À présent, que nous avons fourni le modèle générique pour l'approche PPC, nous indiquons les algorithmes de filtrage que nous avons employés afin de réduire l'effort de recherche.

5.1.4 Propagation des contraintes

Dans la présente section, nous décrivons les filtrages des domaines des variables Sta_k que nous avons intégrés dans le schéma de résolution global. Il est à signaler que les règles de déduction proposées dans cette section sont appliquées à chaque nœud de l'arbre de séparation et d'évaluation. De plus, nous utilisons une approche orientée stations dont le principe est d'ouvrir les stations au fur et à mesure de l'affectation des blocs. Ainsi, la première station ayant l'indice 1 est ouverte. Une fois que l'ensemble des affectations possibles de Sta_1 a été considéré, la seconde station est ouverte et ainsi de suite. Ce processus est réitéré jusqu'à ce que l'ensemble des opérations soit affecté.

À présent, nous introduisons le domaine de chaque ensemble Sta_k pour décrire la propagation des contraintes que nous utilisons, celui-ci est défini comme suit :

$D_{Sta_k} = \{b \in \mathbf{B} \mid k \in [head_b, tail_b]\}$.

où $head_b$ (respectivement $tail_b$) est l'indice de la station au plus tôt (respectivement au plus tard) pour le bloc b tel que $b \in \mathbf{B}$. Nous avons proposé dans la section 5.4 plusieurs algorithmes pour le calcul de ces limites (par défaut $head_b = 1$ et $tail_b = m_0$).

La propagation des contraintes de précédence est appliquée lorsque les deux conditions suivantes sont vérifiées simultanément :

– le bloc b est assigné à Sta_k ;
– le bloc b a des successeurs qui appartiennent au domaine D_{Sta_k}.

Dans ce cas, il est possible d'éliminer l'ensemble des blocs b' appartenant au domaine de la station k qui comportent des opérations successeurs aux opérations de b. Plus formellement, nous appliquons la règle suivante :

$\forall b \in Sta_k$,
Si
$\exists b' \in D_{Sta_k} \mid i \in b, j \in b', (i,j) \in D^{or}$
alors
$D_{Sta_k} := D_{Sta_k} - \{b'\}$

Concernant la propagation des contraintes d'exclusion, nous éliminons un bloc b du domaine D_{Sta_k} lorsqu'un sous-ensemble de $E' \subset E$ tel que $E' = E - \{b\}$ a été affecté à Sta_k. Le bloc b ne doit pas être affecté à la station k car autrement des blocs incompatibles formant E se verraient attribuer une même station. Plus exactement, nous employons la règle suivante :

$\forall E \in D^{ex}$,
Si
$\exists E' \subset \mathbf{B}, \exists b \in \mathbf{B}$ tel que $E' \subseteq Sta_k \wedge E' = E - \{b\}$
alors
$D_{Sta_k} := D_{Sta_k} - \{b\}$

5.1.5 Politique de branchement

Nous avons opté pour un branchement de type : en profondeur d'abord (*DepthFirst*) afin d'obtenir au plus vite des solutions réalisables ce qui fournit une borne supérieure sur l'objectif. L'arbre construit est un arbre n-aire, c'est-à-dire qu'à partir d'un nœud parent plusieurs nœuds fils peuvent être générés.

Avant d'entamer le processus de branchement, il faut construire le graphe de précédence des blocs à partir du graphe de précédence G^{or}. Un branchement correspond à l'affectation d'un bloc disponible à la station courante. La liste de blocs disponibles (noté B_{dispo} dans l'organigramme 5.1) est construite pour chaque nœud de l'arbre car selon les blocs qui ont déjà été affectés de nouveaux blocs deviennent disponibles. Pour obtenir cette liste nous combinons les contraintes de précédence et la propagation des contraintes décrite dans la section précédente.

L'arbre de séparation et d'évaluation est initialisé en fournissant à la racine une borne inférieure sur l'objectif. Un premier niveau de branchements correspond à un point de choix concernant la sélection d'un bloc parmi ceux de la liste des blocs disponibles à la racine. Pour le premier niveau, un branchement revient à affecter un bloc n'ayant pas de prédécesseurs à la première station. Il y aura autant de nœuds fils que de blocs n'ayant pas de prédécesseurs.

Ensuite, pour chaque nœud, une liste des blocs disponibles est construite et le noeud traité devient le noeud parent tel que les fils correspondent à l'affectation des nouveaux blocs disponibles à la station courante. De plus, il faut considérer l'affectation de l'ensemble vide à la station courante au même titre que l'affectation d'un bloc disponible. Ceci signifie que nous mettons fin au chargement de la station courante pour ouvrir une nouvelle station. En effet, cela a pour objectif d'éviter un chargement maximal systématique pour une station. Dans le cas du TLBP, la meilleure solution ne contient pas forcément des stations chargées au maximum car ce sont les blocs les moins chers par opération qui déterminent la solution optimale[3]. De plus, un chargement maximal fait intervenir systématiquement l'ensemble des blocs disponibles ce qui a pour conséquence d'omettre certains blocs alternatifs qui ne seront disponibles qu'à partir des stations ultérieures.

Le processus est réitéré tant qu'il y a des blocs disponibles et que le nombre de stations ouvertes est inférieur à m_0 (le nombre maximum de stations autorisé). Plus exactement, lorsqu'il n'est plus possible d'affecter un bloc à la station en cours, celle-ci est fermée et une autre est ouverte à condition que le nombre de stations ouvertes soit inférieur à la limite m_0. Lorsque celui-ci est supérieure à m_0 l'algorithme s'arrête car le problème est infaisable.

5.1.6 Règle de dominance

Nous avons adapté une règle de dominance d'après les principes d'optimalité de [Bel57]. Cette règle a déjà été appliquée dans [DI05] pour la résolution du TLBP/B-M.

Pour pouvoir appliquer cette règle, il faut définir l'état d'une solution partielle. Celui-ci correspond à l'ensemble des opérations contenues dans les blocs qui ont été déjà affectés.

Proposition 1.

Si pour deux solutions partielles S_1 et S_2, les conditions suivantes sont vérifiées :

- les deux états S_1 et S_2 sont équivalents, c'est-à-dire que les solutions partielles correspondantes contiennent les mêmes opérations ;
- toutes les stations de chacune des solutions partielles sont fermées ;
- Le coût de S_1 est strictement inférieur au coût de S_2.

alors S_1 domine S_2 et cette dernière peut, par conséquent, être éliminée de la recherche.

L'idée de cette proposition vient de la réflexion suivante :

[3]Prenons le cas extrême, s'il y a n opérations et s'il existe un bloc pour chaque opération tel que chacun coûte c et qu'ils peuvent tous être sur une même station. Par ailleurs, s'il existe un bloc contenant toutes les opérations ayant un coût q et que $nc > q$ alors c'est le bloc effectuant toutes les opérations qui déterminera la meilleure solution.

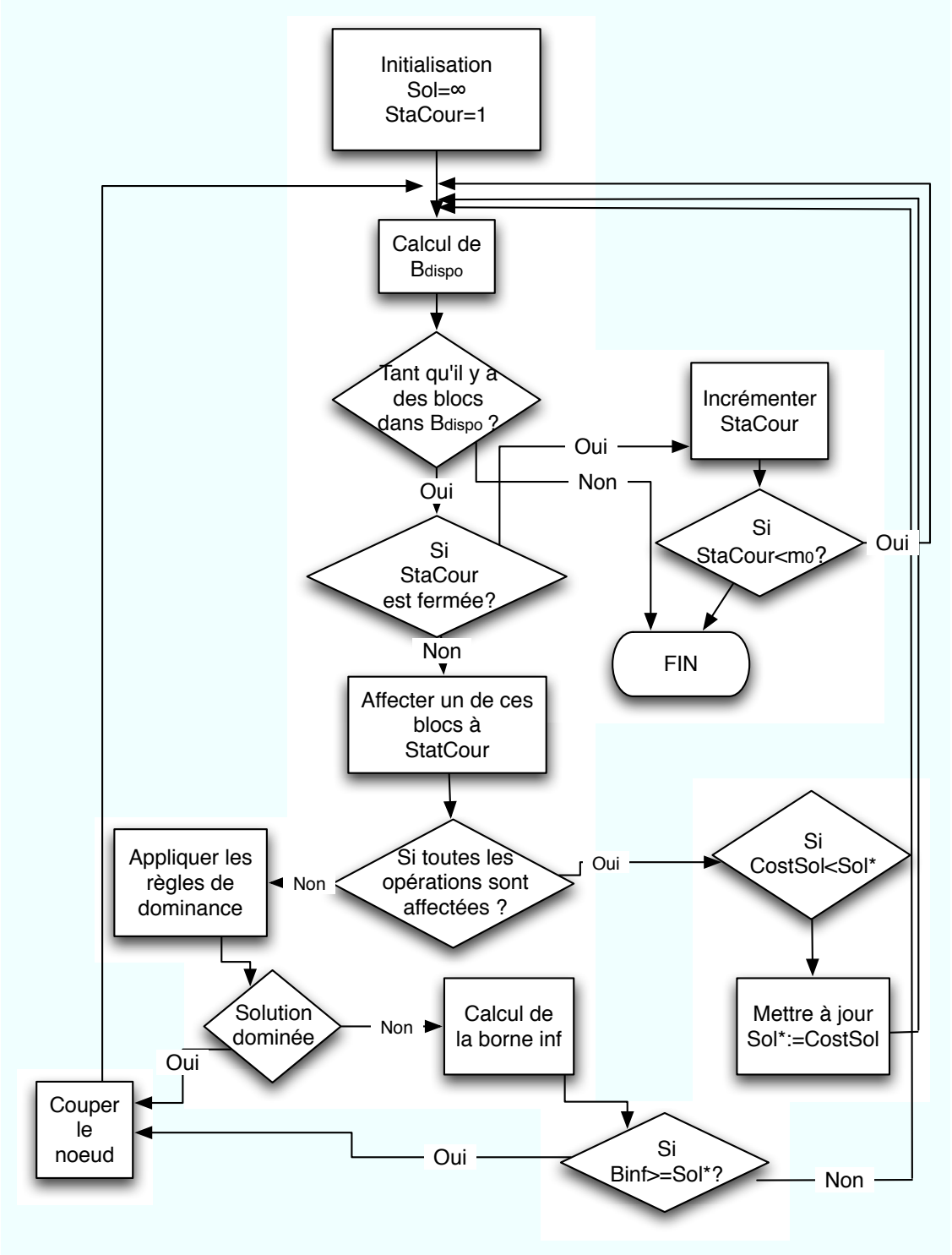


FIG. 5.1 – Schéma de l'approche globale

1. s'il existe deux solutions partielles S_1 et S_2 ayant le même état mais deux coûts différents, c'est-à-dire C_1 et C_2 respectivement, tel que $C_1 < C_2$

2. et si toutes les stations de S_1 et S_2 sont fermées et qu'il existe une sous-séquence optimale de blocs Seq pour compléter une de ces solutions partielles

alors cette sous-séquence est la même pour les deux solutions S_1 et S_2.

De ce fait, les deux solutions réalisables obtenues auront le même rapport de coût que les solutions partielles S_1 et S_2. En d'autres termes, si S_1 domine S_2 alors la solution réalisable composée de S_1 et Seq domine forcément celle qui est composée de S_2 et Seq. Ainsi, toute solution dominée peut être éliminée de la recherche.

Proposition 2.

Lorsque deux solutions partielles S_1 et S_2 ont le même état et qu'elles contiennent les mêmes blocs dans la station courante alors l'une de ces solutions peut être écartée.

Le principe de cette proposition est de détecter les solutions symétriques afin de les éliminer. Par solutions symétriques, nous entendons les solutions considérant les mêmes blocs sélectionnés mais pas la même affectation aux stations.

5.1.7 Approche globale

L'organigramme de l'approche globale est présenté dans la figure 5.1. La procédure commence par calculer l'ensemble des blocs disponibles sur la base des précédences uniquement pour le premier niveau puis en intégrant la propagation des contraintes pour les autres niveaux. Les branchements sont effectués sur la base de cet ensemble de blocs disponibles tel qu'un branchement corresponde à l'affectation de chacun des blocs disponibles à la station courante. Nous appliquons les règles de dominance décrites précédemment avant de calculer la borne inférieure dont le développement est décrit dans la section 5.1.2. L'algorithme s'arrête lorsque tous les nœuds ont été parcourus, la meilleure solution est donnée par Sol^*.

5.2 Un PLNE orienté blocs

Dans cette section, nous proposons une modélisation basée sur la programmation linéaire en nombres entiers. Nous nous sommes d'abord intéressé à une formulation orientée blocs. Nous définissions les variables de décision et présentons le modèle linéaire en nombres entiers ainsi obtenu [DGL$^+$04].

5.2.1 Éléments de modélisation

Pour les précédences

Afin de transcrire les contraintes de précédence qui lient certaines opérations en des contraintes qui lient les blocs qui les contiennent, nous introduisons des ensembles $\Gamma^-(b)$, H, $H(b)$, et une valeur h_{tb} comme suit :

Pour chaque bloc $b \in \mathbf{B}$, nous introduisons un ensemble $\Gamma^{-}(b)$ qui contient les opérations qui précèdent chaque opération appartenant au bloc b, ainsi :

$$\Gamma^{-}(b) = \{i \in \mathbf{N} - \{b\} | \exists j \in b, \quad (i,j) \in D^{or}\}$$

L'ensemble H contient tous les blocs ayant au moins une opération prédécesseur, ainsi :

$$H = \{b \in \mathbf{B} \mid \Gamma^{-}(b) \neq \varnothing\}$$

Pour chaque bloc $b \in \mathbf{B}$, nous calculons l'ensemble $H(b)$ qui contient les blocs effectuant au moins une opérations prédécesseur de b, on écrit alors :

$$H(b) = \{t \in \mathbf{B} \mid t \cap \Gamma^{-}(b) \neq \varnothing\}, \quad \forall b \in \mathbf{B}$$

Le nombre d'opérations prédécesseurs pour les opérations du bloc b qui sont contenues dans le bloc t est donné par :

$$h_{tb} = \mid t \cap \Gamma^{-}(b) \mid, \quad \forall b \in H, \forall t \in H(b)$$

Pour les inclusions

Concernant les relations d'inclusion qui lient certaines opérations, nous introduisons w_{bd} et W_d.

Pour chaque bloc $b \in \mathbf{B}$ et pour chaque sous-ensemble d'opérations $d \in D^{in}$, nous introduisons l'indicateur w_{bd} qui fournit le nombre d'opérations appartenant au bloc b qui sont liées par les contraintes d'inclusion données par d.

$$w_{bd} = \mid \mathcal{N}(b) \cap d \mid, \quad \forall b \in \mathbf{B}, \forall d \in D^{in}$$

Pour chaque élément d'inclusion d, nous introduisons un sous-ensemble W_d qui contient tous les blocs effectuant au moins une opération de d, ainsi :

$$W_d = \{b \in \mathbf{B} \mid w_{bd} > 0\}, \quad \forall d \in D^{in}$$

À chaque opération $i \in d$ correspond un sous-ensemble U_i de $\mathbf{B}$ qui contient tous les blocs pouvant effectuer l'opération i. Formellement, on écrit :

$$U_i = \{b \in \mathbf{B} \mid i \in \mathcal{N}(b)\}, \quad \forall i \in d, \forall d \in D^{in}$$

Pré-traitements spécifiques

À présent, nous suggérons quelques pré-traitements qui exploitent la spécificité des données du problème afin de réduire sa taille. Cette analyse préliminaire a pour objectif de réduire le nombre total de contraintes. Ainsi, entre autres nous distinguons les blocs qui feront l'objet d'un choix de ceux qui seront sélectionnés de façon certaine.

Nous dissocions, parmi l'ensemble $\mathbf{B}$ deux types de blocs :

- Les blocs qui ont des opérations en commun, c'est-à-dire ceux dont l'intersection est non vide. Ces blocs représentent des choix possibles pour chacune des opérations en commun. Une solution réalisable du problème ne doit contenir qu'un seul bloc parmi ces derniers.
- Les blocs qui n'ont pas d'opérations communes. Ces derniers sont donc l'unique possibilité pour effectuer les opérations qui leur appartiennent. Ainsi, ils participent forcément à toute solution réalisable. Formellement, on écrit :
  1. F est un ensemble de sous-ensembles de $\mathbf{B}$ contenant les blocs qui ont au moins une opération en commun. $F = \{F_s, s = 1, \ldots, v\}$ tel que :
     - $s \in \{1, \ldots, v\}, \forall b, b' \in F_s$, $b \cap b' \neq \varnothing$;
     - $\forall (b, b') \in \mathbf{B} \times \mathbf{B}$, si $\mathcal{N}(b) \cap \mathcal{N}(b') \neq \varnothing$ alors $\exists s \in \{1, \ldots, v\}$ tel que $b, b' \in F_s$;
  2. F_0 est le sous-ensemble de $\mathbf{B}$ qui contient les blocs qui n'ont aucune intersection avec d'autres blocs. Par conséquent, si pour une opération donnée $i \in \mathbf{N}$ il n'y a que le bloc b qui la contient alors b appartient forcément à F_0.

- Nous introduisons le sous-ensemble B^* de $\mathbf{B}$ qui contient les blocs qui peuvent être affectés à la dernière station de la ligne.
- m^* est une borne inférieure sur le nombre de stations à ouvrir. Celle-ci est au moins égale à la longueur plus un du plus long chemin dans le graphe de précédence G^{or}. Du fait que les opérations qui sont exécutées sur une même station sont effectuées simultanément, il en résulte que la longueur du plus long chemin du graphe de précédence plus un détermine un nombre minimal de stations à établir[4]. Nous fournissons à la fin de ce chapitre plusieurs algorithmes qui permettent de calculer cette borne.

5.2.2 Les variables de décision

La résolution du problème d'optimisation apporte deux informations pour pouvoir décider de la structure d'une ligne : d'une part il fournit le nombre de stations à établir et d'autre part il détermine l'ensemble des unités d'usinage à mettre en place en précisant leur affectation aux stations. Ainsi, nous définissons les variables de décision correspondantes comme suit :

$$x_{bk} = \begin{cases} 1 & \text{si un bloc } b \in \mathbf{B} \text{ est affecté à la station } k \\ 0 & \text{sinon} \end{cases}, \quad \forall k = 1, \ldots, m_0$$

[4]Ce qui revient à initialiser m^* au rang maximum dans le graphe de précédence, sachant que le rang d'une opération est déterminé par le maximum des rangs de ces prédécesseurs plus un. Il est égal à 1 si l'opération n'a pas de prédécesseur. Nous reviendrons ultérieurement sur le calcul des rangs dans un graphe lors de l'introduction des stations au plus tôt.

La variable de décision y représente le nombre de stations établies.

5.2.3 Formulation des contraintes et objectif

L'objectif est de minimiser le coût total de la ligne. Ce coût est composé du coût engendré par la mise en place de toutes les stations (C étant le coût d'une station), ce qui correspond au premier terme de (5.17), et du coût des unités d'usinage, ce qui est égal au second terme.

$$Minimiser \quad Cy + \sum_{b \in \mathbf{B}} \sum_{k=1}^{m_0} q_b x_{bk} \tag{5.17}$$

Les contraintes (5.18) forcent la sélection des blocs qui n'ont pas d'alternative. Cependant, chacun de ces blocs ne peut être affecté qu'une seule fois.

$$\sum_{k=1}^{m_0} x_{bk} = 1, \quad \forall b \in \mathbf{B} \mid b \in F_0 \tag{5.18}$$

Les contraintes données par (5.19) interdisent l'affectation de plus d'un bloc parmi ceux qui effectuent les mêmes opérations.

$$\sum_{b \in F_s} \sum_{k=1}^{m_0} x_{bk} = 1, \quad \forall s \in \{1, \ldots, v\} \tag{5.19}$$

L'équation (5.20) impose que le nombre total des opérations contenues dans les blocs sélectionnés soit égal au nombre d'opérations qui doivent être effectuées, soit $|\mathbf{N}|$.

$$\sum_{b \in \mathbf{B}} \sum_{k=1}^{m_0} |b| x_{bk} = |\mathbf{N}| \tag{5.20}$$

Afin d'assurer le respect des contraintes de précédence, il suffit d'imposer pour chaque bloc b qui a au moins un prédécesseur, que le nombre d'opérations affectées à des stations strictement antérieures[5] soit supérieur ou égale au nombre d'opérations prédécesseurs du bloc b. Ainsi, c'est l'ensemble des contraintes (5.18), (5.19), (5.20) et (5.21) qui imposent conjointement que toutes les opérations prédécesseurs soient affectées avant les opérations successeurs.

$$\sum_{t \in H(b)} \sum_{s=1}^{s<k} h_{tb} x_{ts} \geq |\Gamma^-(b)| x_{bk}, \quad \forall b \in H, \forall k = 2, \ldots, m_0 \tag{5.21}$$

[5]Une station strictement antérieure est une station qui a un indice strictement inférieur à celui de la station courante.

Pour chaque opération i d'un sous-ensemble d'inclusion $d \in D^{in}$ et pour chaque station potentiellement ouvrable k, nous imposons les équations dictées par (5.22). La partie gauche des contraintes détermine le nombre d'opérations qui appartiennent au sous-ensemble d'inclusion d et qui sont affectées à la station k. Quant à la partie droite, elle exprime le positionnement d'une certaine opération i dans la station k imposant, dans le cas de son affectation, que toutes les autres opérations de ce sous-ensemble soient également assignées à cette station :

$$\sum_{b \in W_d} w_{bd} x_{bk} = |d| \sum_{s \in U_i} x_{sk}, \quad \forall i \in d, \forall d \in D^{in}, \forall k = 1, \ldots, m_0 \tag{5.22}$$

Les inéquations données par (5.23) interdisent l'affectation de l'intégralité des blocs d'un sous-ensemble $E \in D^{ex}$ sur la même station k. Ces contraintes proscrivent le positionnement simultané des variables de décision pour l'affectation de tous les éléments de E à même la station k. Toutefois, nous permettons une affectation des sous-ensembles strictement inclus dans E, c'est-à-dire les sous-ensembles qui ont une cardinalité inférieure ou égale à $|E| - 1$, soit :

$$\sum_{b \in E} x_{bk} \leq |E| - 1, \quad \forall E \in D^{ex}, \forall k = 1, \ldots, m_0 \tag{5.23}$$

Les limitations sur le nombre de blocs dans les stations sont exprimées comme suit :

$$\sum_{b \in \mathbf{B}} x_{bk} \leq n_0, \quad k = 1, \ldots, m_0 \tag{5.24}$$

La valeur de la variable y (nombre de stations ouvertes) est déterminée par le plus grand indice de station parmi celles qui sont ouvertes.

$$y \geq k x_{bk}, \quad b \in \mathbf{B}^*, k = m^* + 1, \ldots, m_0 \tag{5.25}$$

m^* est la borne inférieure sur le nombre de stations. De ce fait, toute solution engendre sûrement un coût de $m^* \times C$ nous pouvons donc considérer les stations qu'à partir de $m^* + 1$.

Les contraintes ont pour but de considérer le nombre de stations ouvertes dans la fonction objectif. Il est à signaler que la variable y peut être considérée comme variable réelle au vu de l'objectif 5.17 et des contraintes (5.25). Dans ce cas, la formulation devient en variables mixtes.

Dans la section suivante, nous intégrons des limites sur les indices des stations pour chacun des blocs afin de réduire le nombre d'affectations possibles. Nous présentons, dans ce qui suit, la reformulation des contraintes. Quant aux algorithmes calculant ses limites, nous les présentons en fin de chapitre car ils sont valables pour tous les modèles proposés.

5.2.4 Intégration des limites dans le modèle linéaire

Afin de réduire le nombre de contraintes et la taille du modèle précédemment décrit nous réduisons l'intervalle d'indices des stations de chaque blocs $\forall b \in \mathbf{B}$ initialement donné par $[1, m_0]$ à un intervalle $K(r) = [head_r, tail_r]$. Les variables de décisions pour chaque blocs $b \in \mathbf{B}$ sont crées uniquement pour les valeurs comprises dans l'intervalle $K(r)$. De manière formelle nous écrivons :

$$x_{bk} = \begin{cases} 1 & \text{si un bloc } b \in \mathbf{B} \text{ est affecté à la station } k \\ 0 & \text{sinon} \end{cases}, \quad \forall k = head_r, \ldots, tail_r$$

Les modifications dues à la réduction de cet intervalle sont données par :

Pour la fonction objectif, il suffit de prendre, pour chaque bloc b, les indices des stations correspondant à l'intervalle $K(b)$:

$$Minimiser \quad Cy + \sum_{b \in \mathbf{B}} \sum_{k=head_b}^{tail_b} q_b x_{bk} \tag{5.26}$$

Pour la reformulation des contraintes (5.18), (5.19) et (5.20) il faut utiliser, pour chaque contrainte, uniquement les variables qui correspondent à l'intervalle $K(b)$:

$$\sum_{k \in K(b)} x_{bk} = 1, \quad \forall b \in F_0 \tag{5.27}$$

$$\sum_{b \in F_s} \sum_{k \in K(b)} x_{bk} = 1, \quad \forall s \in \{1, \ldots, v\} \tag{5.28}$$

$$\sum_{b \in \mathbf{B}} \sum_{k \in K(b)} |b| x_{bk} = |\mathbf{N}| \tag{5.29}$$

Les contraintes de précédence sont exprimées pour chaque station k potentiellement ouvrable[6] et pour chaque bloc ayant au moins un prédécesseur. Ainsi, pour chaque station k nous imposons l'affectation de chaque bloc prédécesseur t dans une station antérieure, c'est-à-dire celle avec un indice s strictement inférieur à k. Toutefois, il faut vérifier que cet indice soit compris dans l'intervalle $K(t) = [head_t, tail_t]$, c'est-à-dire $s \leq tail_t$, or $s < k$ donc $s < min(k, tail_t + 1)$.

$$\sum_{t \in H(b)} \sum_{s=head_t}^{s<min(k,tail_t+1)} h_{tb} x_{ts} \geq |\Gamma^-(b)| x_{bk}, \quad \forall b \in H, \forall k \in K(b) \tag{5.30}$$

[6]Une station potentiellement ouvrable pour un bloc est une station dans laquelle ce bloc peut être assigné, c'est-à-dire une station dont l'indice est compris dans l'intervalle $K(b)$.

Concernant les contraintes d'inclusion, pour chaque opération $i \in d \in D^{in}$, on doit considérer l'union des intervalles de tous les blocs contenant i, car aucune possibilité d'affectation d'un de ces blocs ne doit être omise.

$$\sum_{b \in W_d} w_{bd} x_{bk} = |d| \sum_{s \in U_i} x_{sk}, \quad \forall i \in d, \forall d \in D^{in}, \forall k \in \bigcup_{s \in U_i} K(s) \tag{5.31}$$

En ce qui concerne les contraintes d'exclusion, comme elles sont formulées pour interdire l'affectation de certains blocs à une même station, il suffit de les reformuler uniquement pour les stations pour lesquelles ces blocs sont candidats c'est-à-dire pour les intervalles correspondants.

$$\sum_{b \in E} x_{bk} \leq |E| - 1, \quad \forall E \in D^{ex}, \forall k \in \bigcap_{s \in E} K(s) \tag{5.32}$$

Pour les contraintes limitant la capacité des stations, il suffit de prendre, pour chacune des stations, les blocs qui peuvent lui être affectés. Nous les formulons comme suit :

$$\sum_{b \in \{b' \in \mathbf{B} | k \in K(b')\}} x_{bk} \leq n_0, \quad k = 1, 2, \ldots, m_0 \tag{5.33}$$

Pour les contraintes relatives au nombre de stations ouvertes y, il suffit de retenir, pour chaque station k, que les contraintes sur les variables d'affectation des blocs qui peuvent lui être assignés.

$$y \geq k x_{bk}, \quad b \in \mathbf{B}^*, k \in K(b), k > m^* \tag{5.34}$$

Du fait que m^* est une borne inférieure sur le nombre de stations le coût m^*C est assurément inclus dans le coût de toute solution nous pouvons ainsi démarrer de $m^* + 1$ pour ces contraintes.

5.2.5 Réduction de l'arbre d'énumération

L'idée de cette sous-section a pour objectif de réduire l'effort de calcul en favorisant certaines structures parmi celles qui ont le même coût. Plus précisément, nous avons introduit un ε, qui est suffisamment petit pour ne pas affecter l'ordre global donné par le coûts des blocs[7], toutefois il modifie légèrement les coûts afin de faire la différence entre deux affectations identiques du point de vue du coût de la solution finale mais dont les indices des stations non vides sont distincts. La fonction objectif, telle qu'elle a été exprimée dans les sections précédentes, octroie le même coût à l'affectation d'un bloc, et ce à n'importe quelle station. À présent, nous introduisons le ε pour favoriser l'affectation des blocs aux stations ayant les plus petits indices.

[7]En fait l'ordre entre les coûts des blocs avant et après l'ajout des ε doit être exactement le même.

Il en résulte que le coût engendré par l'affectation d'un bloc augmente proportionnellement par rapport à l'indice de la station à laquelle il est assigné. De cette façon, on pénalise les affectations aux stations ayant les plus grands indices donc la charge des stations de sera pas forcément équilibré. Ceci permet de pénaliser les solutions contenant des stations vides entre deux stations qui ne le sont pas. Plus exactement, ceci évite d'ouvrir une station ayant l'indice $k+1$ sans avoir ouvert celle avec l'indice k. Notre nouvel objectif s'exprime alors comme suit :

$$Minimiser \quad Cy + \sum_{b \in \mathbf{B}} \sum_{k=head_b}^{tail_b} (q_b + k.\varepsilon) x_{bk} \tag{5.35}$$

Cette modification permettra, *a priori*, un gain de temps de calcul mais elle favorise le chargement des stations ayant les petits indices. Mais contrairement aux lignes manuelles, où il est important de lisser la charge, pour les lignes complètement automatisées, il n'est ni nécessaire ni pertinent de chercher à équilibrer la charge de travail. Ainsi nous avons choisi cette structure comme nous aurions pu choisir la structure inverse où les blocs sont plutôt calés à droite.

Le nombre total des solutions identiques peut être très important surtout lorsque le nombre de blocs et m_0 croîent. Dans ce cas, le gain en temps de calcul qui peut être obtenu en utilisant cette modification est potentiellement important.

Dans la suite, nous proposons une seconde formulation en nombres entiers dans le but d'améliorer les performances du modèle précédent. L'idée apportée dans cette nouvelle formulation est de raisonner non plus en fonction des blocs d'opérations mais plutôt en fonction des opérations elles-mêmes quand il s'agit de contraintes qui leur sont intrinsèques, telles que les contraintes de précédence et d'inclusion [BDGL05b].

5.3 Un PLNE orienté opérations

Dans cette section, nous présentons les éléments de modélisation que nous avons utilisés pour exprimer les contraintes du point de vue des opérations (à l'exception des contraintes qui sont intrinsèques aux blocs telles que les contraintes d'exclusion ou des contraintes sur le nombre maximum de blocs par station).

5.3.1 Les éléments de modélisation

Pour chaque opération $i \in \mathbf{N}$ nous introduisons $Q(i)$ désignant l'ensemble des unités d'usinage qui peuvent effectuer l'opération i. Plus formellement, on écrit :

$Q(i) = \{b \in \mathbf{B} | i \in \mathcal{N}(b)\}, \forall i \in \mathbf{N}$

Ainsi, une décision réalisable comprend pour chaque opération i un seul bloc de $Q(i)$.

Nous utilisons dans cette seconde formulation les stations au plus tôt et les stations au plus tard des blocs afin de réduire le nombre de variables binaires x_{rk}.

Les intervalles d'indices des stations pour les opérations sont employés pour formuler directement les contraintes les concernant, ils sont notés $KO(j)$ pour toute opération j. Ces intervalles sont déduits à partir des intervalles d'affectation des blocs, soit :

$$KO(j) = \bigcup_{b \in \{b' | j \in \mathcal{N}(b')\}} K(b), \forall j \in \mathbf{N}$$

5.3.2 Les variables de décision

Nous réutilisons les mêmes variables concernant les décisions relatives à l'affectation des blocs aux stations, soit :

$$x_{bk} = \begin{cases} 1, & \text{si le bloc } b \text{ est affecté à la station } k, \\ 0, & \text{sinon} \end{cases} , \forall b \in \mathbf{B}, \forall k = head_r, \ldots, tail_r$$

Toutefois, pour la décision relative à l'ouverture d'une station, nous introduisons les variables binaires suivantes :

$$y_k = \begin{cases} 1, & \text{si la station } k \text{ est créée (ouverte),} \\ 0, & \text{sinon} \end{cases} , \forall k = 1, \ldots, m_0$$

5.3.3 La formulation des contraintes et objectif

L'objectif est exprimé par l'équation (5.36). De la même façon que dans la première formulation, la double somme sur les variables binaires x_{bk} correspond aux coûts engendrés par les unités d'usinage alors que la somme sur les variables y_k représente le coût associé à l'ouverture des stations.

$$Minimiser \sum_{k=m^*+1}^{m_0} C y_k + \sum_{b \in \mathbf{B}} \sum_{k=head_b}^{tail_b} q_b x_{bk} \tag{5.36}$$

Comme nous l'avons déjà évoqué, le coût m^*C est induit dans toute solution réalisable du fait que m^* est une borne inférieure sur le nombre de stations, nous pouvons de ce fait formuler l'objectif en considérant uniquement les stations postérieurs à m^*.

Lors de la précédente formulation nous en avions besoin des différentes contraintes (5.18), (5.19) et (5.20) afin de modéliser l'unicité de l'exécution des opérations. A présent, il suffit d'imposer à chacune des opérations qu'il n'y ait qu'un seul bloc qui la contienne qui soit sélectionné :

$$\sum_{b \in Q(i)} \sum_{k=head_b}^{tail_b} x_{bk} = 1, \quad \forall i \in \mathbf{N} \tag{5.37}$$

L'inéquation (5.38) est construite pour chaque couple d'opérations appartenant à D^{or}. Nous avons, en ce sens, simplifié leur modélisation car dans le modèle précédent une contrainte a été exprimée pour chaque bloc b ayant au moins un prédécesseur. À présent, au lieu de raisonner en fonction de chaque bloc (voir 5.30), nous traduisons la formulation pour tous les couples d'opérations (i, j). En fait, il suffit d'utiliser les ensembles $Q(i)$ et $Q(j)$ et d'imposer l'ordre d'exécution décrit par la contrainte (i, j), soit i avant j. Ce qui revient à forcer les blocs effectuant j à être assignés à une station strictement postérieure[8] à celle qui effectuera l'opération i.

$$\sum_{b \in Q(i)} \sum_{l=head_b}^{min(k-1, tail_b)} x_{bl} \geq \sum_{b \in Q(j)} x_{bk}, \quad \forall (i,j) \in D^{or}, \forall k \in KO(j) \tag{5.38}$$

Pour modéliser les contraintes d'inclusion nous formulons une contrainte pour toute paire d'opérations (i, j) appartenant au même sous-ensemble d'inclusion et ceci pour chaque station possible, comme indiqué dans (5.39). La partie droite de l'équation correspond à l'affectation de l'opération j dans la station k. Pour que ces contraintes soient respectées nous imposons aux variables déterminant l'affectation de l'opération i à être égales à celles de j. Ainsi, si un des deux opérations est affectée à la station k alors la seconde doit l'être également et vice versa[9].

$$\sum_{b \in Q(i)} x_{bk} = \sum_{r \in Q(j)} x_{rk}, \quad \forall (i,j) \subseteq d, \forall d \in D^{in}, \forall k \in KO(j) \tag{5.39}$$

Du fait que par définition les contraintes d'exclusion sont intrinsèques aux blocs, nous réutilisons la formulation introduite dans le modèle précédent (orienté bloc).

$$\sum_{b \in E} x_{bk} \leq |E| - 1, \quad \forall E \in D^{ex}, \quad \forall k \in \bigcap_{b' \in E} K(b') \tag{5.40}$$

Chaque station qui peut être ouverte ne doit pas contenir plus de n_0 blocs. Ainsi, après l'introduction des variables relatives à l'ouverture d'une station, ces contraintes sont exprimées au moyen de (5.33).

$$\sum_{b \in \mathbf{B}} x_{bk} \leq n_0, \quad k = 1, \ldots, m_0 \tag{5.41}$$

[8]Car les indices des stations sont numérotées de 1 à m_0.

[9]Comme l'ordre de l'affectation des blocs et par conséquent des opérations n'est pas prise en compte dans la formulation, nous passons outre le fait que l'opération i soit affecté avant j ou que j le soit avant. En fait l'équation (5.39) va forcer les variables correspondantes à l'affectation de l'opération qui n'est pas encore affectée à être égales à celles de l'opération qui l'a déjà été

Dés lors qu'un bloc est affecté à une station, la variable indiquant l'ouverture de cette station est positionnée à un en ajoutant les contraintes suivantes :

$$y_k \geq x_{bk}, \quad \forall k \geq m^* + 1, \forall b \in \{r \mid r \in \mathbf{B}, k \in K(r)\} \tag{5.42}$$

Nous utilisons les contraintes décrites par (5.43) afin de générer des solutions dans lesquelles les stations sont créées dans l'ordre lexicographique, allant de 1 à m_0. Ainsi, dans une solution si la première station est saturée[10] alors la prochaine station à ouvrir aura l'indice suivant, soit égale à 2 et ainsi de suite. En fait, en l'absence de ces contraintes, l'indice de la prochaine station à ouvrir aura une valeur arbitraire comprise entre 1 et m_0 à l'exception des indices des stations qui ont déjà été ouvertes. Ces contraintes jouent le même rôle que la technique utilisant une constante ε dans l'objectif (voir l'expression (5.35)).

$$y_{k-1} - y_k \geq 0, \quad k = m^* + 2, \ldots, m_0 \tag{5.43}$$

5.4 Réduction du nombre de variables

Cette étape consiste à réduire, pour chaque bloc, l'intervalle définissant la plage d'indices des stations dans lesquelles ce bloc peut être affecté. Dans le modèle (5.17)-(5.25) nous avons considéré pour chacun des blocs toutes les affectations possibles (c'est-à-dire toutes les stations potentiellement ouvrables). Ainsi, cet intervalle est égale initialement à $[1, m_0]$. Nous pouvons pourtant affiner cet intervalle en nous basant sur certaines contraintes qui vont interdire certaines affectations. C'est pourquoi nous avons introduit la notion d'intervalle réduit $K(b)$ pour chaque bloc $b \in \mathbf{B}$.

L'idée que nous proposons à présent est d'étudier la consistance aux bornes de ces intervalles $K(b) = [head_b, tail_b]$ tel que $head_b \geq 1$ et que $tail_b \leq m_0$. L'objectif est de réduire au maximum pour chaque bloc b l'intervalle des indices de stations possible à un intervalle. Nous proposons d'appliquer un algorithme qui exploite la structure des contraintes pour déduire des impossibilités d'affectations des blocs dans certaines stations. De ce fait, nous détectons les indices des stations qui mèneront à des solutions non réalisables, ce qui nous permet de réduire les intervalles et d'éliminer ensuite les variables binaires correspondantes à ces affectations[11].

Pour ce faire, nous proposons trois algorithmes, le premier exploite uniquement les contraintes de précédence tandis que le second intègre en plus les contraintes d'inclusion pour affiner les limites des intervalles $K(b)$. Le dernier algorithme est plus complexe dans le sens où il fait intervenir l'ensemble des contraintes.

[10] C'est-à-dire qu'il devient impossible de lui affecter un bloc supplémentaire sans la violation d'une des contraintes.

[11] Ces solutions engendrées par l'affectation des blocs aux stations qui violent certaines contraintes sont écartées et permettent ainsi de réduire l'espace de recherche

5.4.1 Algorithme 1

Nous introduisons le terme de «station au plus tôt», pour désigner la limite $head_b$ du bloc b et «station au plus tard», pour la limite $tail_b$. Les valeurs contenues dans cet intervalle représentent les indices des stations que nous allons considérer pour la création de variables de décision correspondantes. Par souci de clarté, nous employons le terme limite pour tout $head_b$ ou $tail_b$, au lieu du terme borne afin de ne pas créer d'ambiguïté due à l'association du mot borne à la borne inférieure sur la fonction objectif ou sur le nombre de stations.

Le principe de cet algorithme est d'exploiter les contraintes de précédence qui existent entre les opérations pour en déduire des limites pour les blocs qui les contiennent. Par exemple, s'il y a une opération qui doit précéder une opération appartenant au bloc b, alors le bloc successeur ne pourra pas être affecté à la première station, son intervalle peut donc commencer à partir de l'indice 2 indiquant la deuxième station.

Cet algorithme a une complexité polynomiale ce qui le rend très intéressant d'un point de vue pratique.

Rappelons qu'afin d'obtenir les rangs des opérations il faut appliquer ce qui suit :

$$rank(j) = \begin{cases} \max\{rank(i) \mid i \in \mathbf{N}, (i,j) \in G^{or}\} + 1, \text{ si } j \text{ a des prédécesseurs} \\ 1 \text{ sinon.} \end{cases}$$

Le rang d'un bloc est déterminé par le maximum des rangs des opérations qui le composent. La station au plus tôt d'un bloc b correspond à son rang dans le graphe de précédence.

Algorithme 1
Begin

Calculer les rangs des opérations

For all $b \in \mathbf{B}$
$head_b := \mathbf{max}\ \{rank(j), j \in \mathcal{N}(b)\}$
End For

End

Pour obtenir les stations au plus tard, il faut d'abord inverser le graphe de précédence G^{or}, on obtient un graphe G^{or}_{inv}. Ensuite, il faut calculer les rangs des opérations dans ce graphe en appliquant la règle suivante :

$$\overline{rank}(j) = \begin{cases} \min\{\overline{rank}(i) \mid i \in \mathbf{N}, (i,j) \in G^{or}_{inv}\} - 1; \text{ si } j \text{ a un prédécesseur} \\ m_0, \text{ sinon} \end{cases}$$

Ensuite, pour obtenir les $tail_b$ pour chaque bloc b, il suffit de remplacer dans l'Algorithme 1 l'instruction calculant les *head* par la suivante :

$tail_b := \mathbf{min}\{\overline{rank}(j), \forall j \in \mathcal{N}(b)\}$

Un exemple

Nous reprenons l'exemple que nous avons introduit dans la section 4.2 pour appliquer l'Algorithme 1. Nous rapportons les valeurs des rangs dans le tableau 5.1 et celles des stations au plus tôt et au plus tard dans le tableau 5.2.

op	$rank(j)$	$\overline{rank}(j)$	op	$rank(j)$	$\overline{rank}(j)$
1	1	$m_0 - 1$	8	3	$m_0 - 1$
2	1	$m_0 - 1$	9	1	$m_0 - 2$
3	1	$m_0 - 3$	10	2	$m_0 - 1$
4	2	$m_0 - 2$	11	4	m_0
5	3	$m_0 - 1$	12	1	$m_0 - 2$
6	1	$m_0 - 3$	13	2	$m_0 - 1$
7	2	$m_0 - 2$	14	4	m_0

TAB. 5.1 – Les rangs des opérations

bloc	opérations	$head_b$	$tail_b$	bloc	opérations	$head_b$	$tail_b$
b_1	1,3,6	1	$m_0 - 3$	b_{12}	1,2,4,7	1	$m_0 - 3$
b_2	2,4,7	2	$m_0 - 2$	b_{13}	3,6	1	$m_0 - 3$
b_3	1,2,3,6	1	$m_0 - 3$	b_{14}	1,6	1	$m_0 - 3$
b_4	4,7	2	$m_0 - 2$	b_{15}	1	1	$m_0 - 1$
b_5	9,12	1	$m_0 - 2$	b_{16}	2	1	$m_0 - 1$
b_6	10,13	2	$m_0 - 1$	b_{17}	1,2	1	$m_0 - 1$
b_7	5,8	3	$m_0 - 1$	b_{18}	3	1	$m_0 - 3$
b_8	11	4	m_0	b_{19}	6	1	$m_0 - 3$
b_9	14	4	m_0	b_{20}	5	3	$m_0 - 1$
b_{10}	2,3,6	1	$m_0 - 3$	b_{21}	8	3	$m_0 - 1$
b_{11}	1,4,7	2	$m_0 - 2$				

TAB. 5.2 – Les indices de station au plus tôt et au plus tard

5.4.2 Algorithme 2

Nous introduisons un second algorithme pour le calcul des stations au plus tôt et au plus tard. Dans cette version, nous exploitons non seulement les contraintes de précédence mais également les contraintes d'inclusion. De façon similaire à l'Algorithme 1, cet algorithme est d'abord utilisé pour l'obtention des stations au plus tôt $head_b$ puis il est adapté, après quelques substitutions, pour le calcul des stations au plus tard $tail_b$.

La première étape de l'algorithme consiste à initialiser, pour chaque opération j de $\mathbf{N}$ sa limite $ub(j)$ avec la valeur correspondante à son rang dans le graphe de précédence.

La variable n_{imp} est également initialisée, elle sert à détecter la présence éventuelle d'un cycle auquel cas l'algorithme est terminé en indiquant l'absence de solutions réalisables.

Algorithme 2

Begin

Step1. **For all** $(j \in \mathbf{N})$
$ub(j) := rank(j)\,;$
$n_{imp} := 0\,;$

Step2. Set $imp_{ub} := 0\,;$

Step3. **For all** $(d \in D^{in})$
$um := max\{ub(j) \mid j \in d\}\,;$
For all$(j \in d)$
$ub(j) := um\,;$

Step4. **For all** $(j \in \mathbf{N})$
$u_j := max\{ub(i) + 1 \mid, \forall i \in \mathbf{N}, (i,j) \in D^{or}\}\,;$
If$(u_j > ub(j))$ **Then**
$ub(j) := u_j\,;$
$imp_{ub} := imp_{ub} + 1\,;$
EndIf

Step5. Set $n_{imp} := n_{imp} + 1\,;$
If$(n_{imp} > |\mathbf{N}|)$ **Then**
stop (il n'y a pas de solutions réalisables)
EndIf
If$(imp_{ub} > 0)$ **Then**
goto **Step2**;
EndIf

Step6. **For all** $(b \in \mathbf{B}\ )$
Set $head_b := \mathbf{max}\{ub(j) \mid j \in \mathcal{N}(b)\}\,;$

Step7. Set $m^* := \mathbf{max}\{ub(j) \mid j \in \mathbf{N}\}\,;$

End

Quant à la variable imp_{ub} elle compte le nombre d'améliorations effectuées dans une itération de l'algorithme.

La troisième étape (step3) consiste à ramener le rang de toutes les opérations, pour chaque sous-ensemble d'inclusion d au plus grand rang dans d. En effet, par définition, les opérations de chaque ensemble d doivent être affectées à la même station. L'étape 4 consiste à obtenir les limites des stations pour les opérations en se basant sur les contraintes de précédence. L'étape 5 consiste à détecter une amélioration des limites auquel cas il faut retourner à l'étape 2. Lorsqu'il y a plus de $|\mathbf{N}|$ améliorations le graphe de précédence contient un cycle et le problème correspondant est insolvable. L'étape 6 permet de déduire les limites pour les blocs à partir de celles pour les opérations. L'étape 7, permet d'obtenir une borne inférieure sur le nombre de stations. Elle est obtenue simplement en calculant le maximum des limites de l'ensemble des opérations (ou blocs).

L'amélioration par rapport au précédent algorithme d'imposer aux blocs ayant des opé-

rations liées par des contraintes d'inclusions d'avoir les mêmes limites. En d'autres termes, si un bloc b contient une opération i et un bloc b' contient une autre opération j tel que (i, j) appartiennent à D^{in} alors b et b' doivent avoir le même interval, c'est-à-dire que $K(b) = K(b')$. La complexité reste polynomiale, car au pire des cas, le graphe de précédence est parcourue $|\mathbf{N}|$.

5.4.3 Algorithme 3

Cet algorithme prend en compte toutes les contraintes. Pour l'expliquer nous utilisons la notion de niveau g (voir Algorithme 3). *A priori*, la répartition des opérations en niveaux correspond aux rangs du graphe de précédence mais la prise en compte des autres contraintes peut amener à obtenir plusieurs niveaux par rang. Le principe est d'identifier pour chaque niveau un sous-ensemble maximal de blocs (B^{max}) qui vérifie l'ensemble des contraintes, à savoir : les contraintes de précédence, inclusion, exclusion et n_0 (le nombre maximum de blocs par station). Le sous-ensemble maximal de blocs se voit alors attribuer g comme station au plus tôt. L'algorithme s'arrête lorsque g dépasse le nombre maximum de stations m_0. Aussi, si après la boucle *Repeat while* il subsiste des opérations dont les blocs n'ont pas pu être sélectionnés alors le problème est infaisable.

Comme dans les deux cas précédents, l'algorithme 3 est d'abord utilisé pour l'obtention des stations au plus tôt $head_b$ pour tout bloc b. Il est ensuite adapté pour l'obtention des stations au plus tard $tail_b$ en effectuant les substitutions suivantes :

1. $g = m_0$ au lieu de $g = 1$, à l'étape de l'initialisation
2. $g \geq 1$ au lieu de $g \leq m_0$, dans *Repeat while*
3. $g = g - 1$ au lieu de $g = g + 1$
4. $\Gamma^-(b)$ est remplacé par l'ensemble $i \in \mathbf{N} \backslash \mathcal{N}(b)$ tel que $(j, i) \in D^{or}$ pour tout $j \in \mathcal{N}(b)$
5. $B_1 := B_1 \cup \{b \in \mathbf{B} - B_1 | head_b > tail_b\}$ à mettre après le *End Repeat* ;

L'inconvénient majeur de cet algorithme est incontestablement sa complexité. En effet, la recherche d'un sous-ensemble maximal $B^{max} \subseteq \mathbf{B}_1$ nécessite un parcours total de toutes les combinaisons possibles de sous-ensembles, c'est-à-dire : $\sum_{k=1}^{|\mathbf{B}_1|} C^k_{|\mathbf{B}_1|}$, tel que : $C^k_n = \frac{n!}{k!(n-k)!}$. Cette complexité ne permet pas d'employer l'algorithme de façon efficace. Il est évident qu'il sera moins performant sur des instances de grande taille, c'est pourquoi nous avons opté pour l'implémentation des deux premiers algorithmes uniquement.

Algorithme 3
Begin
$g := 1$; $B_1 := \mathbf{B}$; $B_2 := \varnothing$; $D_0 := D^{in}$;
For all $b \in \mathbf{B}$
$M_1(b) := \Gamma^-(b)$; $M_2(b) := \varnothing$;
Repeat while $(B_1 \neq \varnothing \wedge g \leq m_0)$
$B^{max} = \varnothing$;
For all $(B_{temp} \subseteq B_1)$
$\forall b \in B_{temp}$
IF$\Big(\big(|B_{temp}| > |B^{max}|\big) \bigwedge \big(M_1(b) = \varnothing\big) \bigwedge$
$\big((\forall d \in D_0, w_{bd} = 0) \vee (\exists B' \subseteq B^+, b \in B', d \subseteq \bigcup_{\mathcal{N}(b) \in B'} b, |B'| \leq n_0 \wedge B' \notin D^{ex})\big)$
$\bigwedge \Big((M_2(b) = \varnothing) \vee (\exists B' \subseteq B_2, M_2(b) \subseteq \bigcup_{\mathcal{N}(b) \in B'} b, |B'| \leq n_0, B' \notin D^{ex})\Big)\Big)$
Then $B^{max} := B_{temp}$
End For
If $(B^{max} = \varnothing)$
Then break;
For all $b \in B^{max}$,
$head_b = g$;
End For
$B_1 := B_1 - B^{max}$; $B_2 := B^{max}$; $g := g + 1$;
$D_0 := D_0 - \{d \in D_0 \mid d \subseteq \bigcup_{b \in B^{max}} \mathcal{N}(b)\}$;
For all $b \in B_1$
$M_2(b) := M_1(b) \cap (\bigcup_{b \in B^{max}} \mathcal{N}(b))$;
$M_1(b) := M_1(b) - M_2(b)$;
End For
End Repeat

If$(B_1 \neq \varnothing)$ **Then**
$\mathbf{B} := \mathbf{B} - B_1$;
If$(\bigcup_{b \in \mathbf{B}} \mathcal{N}(b) \neq \mathbf{N})$
Then return **False**
Else
For all $E \in D^{ex}$
If$(E \cap B_1 \neq \varnothing)$
$D^{ex} := D^{ex} - E$;
End If
End If
End If
Return True
End

5.5 Conclusion

Dans ce chapitre, nous avons proposé plusieurs modèles en affinant la formulation des contraintes afin d'exploiter au mieux la structure du problème. Pour la résolution des modèles proposés, nous avons employé les outils d'ILOG. En particulier, nous utilisons Cplex pour la programmation linéaire en nombres entiers et Solver pour la programmation par contraintes. Malgré les performances de ces outils, une bonne modélisation reste un facteur prépondérant quant à l'efficacité de résolution.

Nous avons apporté plusieurs modélisations du problème TLBP/B-P. Nous avons suggéré trois modèles, le premier est générique quant aux deux autres ils sont basés sur la programmation linéaire en nombres entiers.

Dans le cadre de la démarche basée sur la programmation par contraintes, nous avons intégré une borne inférieure basée sur la relaxation linéaire et suggéré également une autre borne obtenue par relaxation en un problème de set partitioning particulier. L'approche s'appuie également sur l'emploi de règles de dominance pour réduire l'espace des solutions énumérées et sur la propagation des contraintes d'exclusion et de précédence.

Pour la programmation linéaire en nombres entiers, nous avons proposé deux formulations : la première est orientée blocs, la seconde est orientée opérations. Le second modèle intègre également des coupes afin de réduire l'espace de recherche en favorisant certaines structures de solutions. Ces coupes ont été introduites pour remplacer la modification introduisant une constante ε dans l'objectif du premier modèle (voir sous-section 5.2.5).

De plus, nous avons fourni plusieurs algorithmes afin de réduire le nombre de variables diminuant ainsi la taille du modèle. Nous soulignons le caractère générique des algorithmes proposés car ils s'appliquent à l'ensemble des modèles présentés.

Dans le chapitre suivant, nous rapportons les résultats des tests expérimentaux. Ainsi, nous évaluons les performances des formulations proposées. De plus, nous y montrons l'apport des améliorations proposées. Enfin, nous retiendrons l'approche la plus performante pour étudier ensuite l'impact de certaines caractéristiques, *a priori*, influentes sur les performances des modèles.

Chapitre 6

Phase expérimentale pour le TLBP/B-P

Dans ce chapitre, nous rapportons les résultats des tests expérimentaux que nous avons effectués pour le problème TLBP/B-P. L'objectif de cette étude est triple, il s'agit d'abord de montrer l'apport des différentes améliorations en particulier celle de la réduction du nombre de variables. Ensuite, nous comparons les différents modèles, et enfin nous étudions l'influence des paramètres du problème. Tous les calculs ont été effectués sur une station de travail de type HP Xeon, avec un processeur de 3,6 GHz et une mémoire vive de 2 Go. Les programmes sont écrits en langage C++ et le compilateur utilisé est Visual Studio version 6.0.

6.1 Description de l'environnement des tests

6.1.1 Le logiciel de résolution

Nous utilisons les logiciels ILOG Cplex et ILOG Solver qui sont des librairies programmées en C++ proposées par ILOG pour la résolution de modèles linéaires en nombres entiers, en variables mixte ou encore de modèles quadratiques. Pour résoudre un PLNE, Cplex met en œuvre un algorithme de *Branch & Cut*, c'est-à-dire une procédure par séparation et évaluation intégrant plusieurs types de coupes telles que les coupes de cliques et les coupes de sac à dos. Quant à Solver c'est une PSE qui intègre des algorithmes de filtrage associés à certaine contraintes du type *allDiff* pour imposer aux variables de prendre des valeurs distinctes.

6.1.2 Jeux de données (instances)

Pour pouvoir faire des conclusions statistiques, nous générons aléatoirement un grand nombre des jeux de données. Pour générer ces instances, nous fixons les paramètres tels que les densités des graphes de contraintes et les cardinalités des ensembles **N** et **B**. Nous y intégrons l'ensemble des pré-traitements décrits précédemment (voir la sous-section 4.3.1). Nous vérifions également la cohérence des paramètres entre eux. Par exemple, si nous générons un

graphe de précédence dont le chemin le plus long est égale à une valeur $k+1$, il n'est pas possible d'imposer une borne supérieure sur le nombre de stations m_0 qui soit inférieure[1] à k . Chaque instance générée est donc réalisable. Nous indiquons dans le tableau 6.1 les paramètres les plus importants que nous avons utilisé pour construire les jeux de données.

Paramètres	descriptif
$OpMin$	le nombre minimum d'opérations par bloc, par défaut cette valeur est à 1
$OpMax$	le nombre maximum d'opérations par bloc, par défaut cette valeur est à 5
$Min\|\mathbf{B}\|$	la cardinalité minimale de l'ensemble $\mathbf{B}$
$Max\|\mathbf{B}\|$	la cardinalité maximale de l'ensemble $\mathbf{B}$
$Avg\|\mathbf{B}\|$	la cardinalité moyenne de l'ensemble $\mathbf{B}$
OS	la densité du graphe de précédence G^{or}
OS_{min}	la densité minimale du graphe G^{or}
OS_{max}	la densité maximale du graphe G^{or}
OS_{avg}	la densité moyenne du graphe G^{or}
OSI	l'intervalle défini par $[OS_{min}, OS_{max}]$

TAB. 6.1 – Descriptif des paramètres pour la génération d'instances d'une famille

Nous utilisons OS comme mesure de la densité du graphe de précédence [Sch99]. Pour la définir, nous introduisons la matrice P telle que tout élément p_{ij} de cette matrice est décrit par :

$$p_{ij} = \begin{cases} 1 & \text{si l'opération } i \text{ précède l'opération } j, \\ 0 & \text{sinon.} \end{cases}$$

La densité du graphe de précédence est alors donnée par :

$$OS = \frac{2z}{|\mathbf{N}|(|\mathbf{N}|-1)} \tag{6.1}$$

où z est le nombre d'éléments non nuls de P. Ce ratio est nul pour un jeu de données qui ne comporte aucune contrainte de précédence. Lorsque l'ordre entre les opérations est complet, c'est-à-dire qu'il y a plus qu'une seule séquence d'opérations possible, alors ce ratio est égal à un.

Il est à souligner que nous avons généré chacune des instances de manière à ce que les densité des différents graphes de contraintes ainsi que leur structure soit les plus proches de celles des instances industrielles dont nous disposons. Par exemple, les contraintes de précédence dans l'usinage (contrairement à l'assemblage) ont une forme de graphe série-parallèle, c'est-à-dire plusieurs séquences d'opérations sont en parallèle telles qu'une séquence possible peut être : perçage d'ébauche, filetage et finition. De façon générale, nous testons 7 familles d'instances que nous décrivons dans le tableau 6.2. Pour chaque famille, nous serons amenés à générer différentes sous-familles en fonction de l'étude effectuée que nous décrirons plus en détails aux sous-sections correspondantes.

[1]Ceci est dû à l'activation parallèle des blocs au sein d'une station, suite à quoi deux opérations liées par une contrainte de précédence ne peuvent pas être affectées simultanément à la même station.

Familles	$\|\mathbf{N}\|$	m_0	$[Min\|\mathbf{B}\|, Max\|\mathbf{B}\|]$	$Avg\|\mathbf{B}\|$
F_{10}	10	6	[13,17]	15
F_{15}	15	8	[20,27]	25
F_{20}	20	13	[29,31]	30
F_{30}	30	16	[45,50]	48
F_{40}	40	21	[57,70]	68
F_{50}	50	26	[67,94]	93
F_{60}	60	30	[117,119]	113
F_{70}	70	34	[125,135]	133
F_{80}	80	38	[144,154]	152

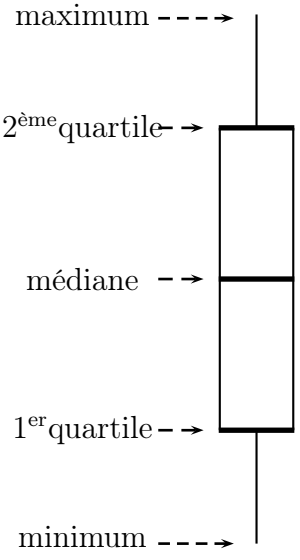
TAB. 6.2 – Les valeurs des paramètres pour la génération des familles instances

6.1.3 Représentation des résultats statistiques

Nous rapportons les résultats des expérimentations globalement sous deux formes. La première forme est classique, elle correspond à un nuage de points représentant chacun le temps de résolution d'une instance donnée appartenant à une des familles. Nous utilisons cette représentation lorsque le nombre d'instances testées le permet.

Pour les sections dont le nombre de tests est très important, nous employons une seconde forme nommée **candlesticks**. Celle-ci représente les temps d'exécution des instances appartenant à une sous-famille donnée. Sa représentation graphiquement nécessite de calculer les valeurs suivantes :

- **le maximum** : le point extrême le plus haut (limite de la droite),
- **le 3ème quartile** : pour un échantillon de valeurs triées, il correspond à la valeur délimitant deux tranches, à savoir : celle des 75% plus petites valeurs des 25% restantes. Cette valeur représente graphiquement le côté supérieur du rectangle,
- **le 1er quartile** : cette valeur délimite les 25% plus petites valeurs de l'échantillon des autres qui leurs sont supérieures, il est indiqué graphiquement par le côté inférieur du rectangle,
- **le minimum** : le point extrême le plus bas. Nous intégrons la médiane de l'échantillon représentée graphiquement par un trait horizontal.



6.2 Évaluation de l'approche PPC

Nous comparons dans cette section l'approche PPC avec le modèle linéaire orienté blocs. Pour ce faire, nous testons les trois premières familles décrites précédemment et indiquons les temps de calcul obtenus dans les figures 6.1, 6.2 et 6.3.

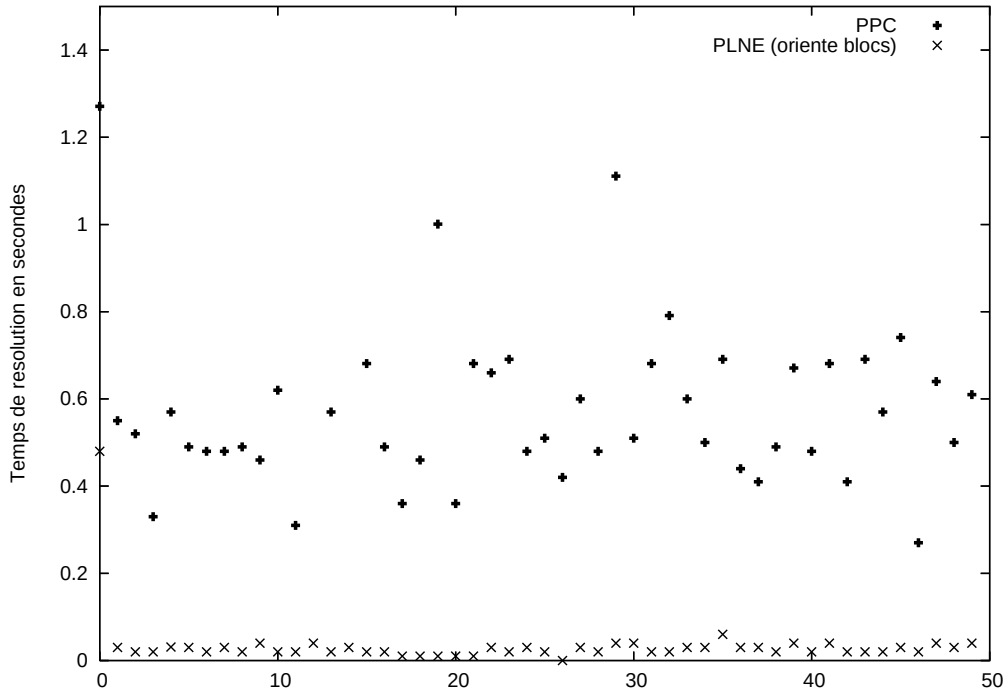


FIG. 6.1 – Temps de calcul des modèles PPC et PLNE pour les instances de F_{10}

Pour l'ensemble des 150 instances testées, le modèle linéaire orienté blocs (5.17)-(5.25) est plus performant. En effet, quand le modèle orienté blocs résout l'ensemble des instances en près d'une seconde, le modèle PPC peut aller jusqu'à 350 sec (voir figures 6.2 et 6.3). De plus, le modèle PPC n'a pas permis de résoudre certaines instances ayant plus de 20 opérations (sous la limite de 2h de temps). Nous signalons toutefois les difficultés rencontrées lors de l'implémentation du modèle PPC qui sont essentiellement dues au manque de documentation sur ILOG Solver d'une part et à l'instabilité du logiciel lui-même lors de la résolution de nombreuses instances d'autre part.

6.3 Modèle orienté blocs

6.3.1 Réduction du nombre de variables

L'objectif de cette sous-section est de faire une analyse comparative de l'efficacité des algorithmes de réduction du nombre de variables. Ces algorithmes calculent les intervalles

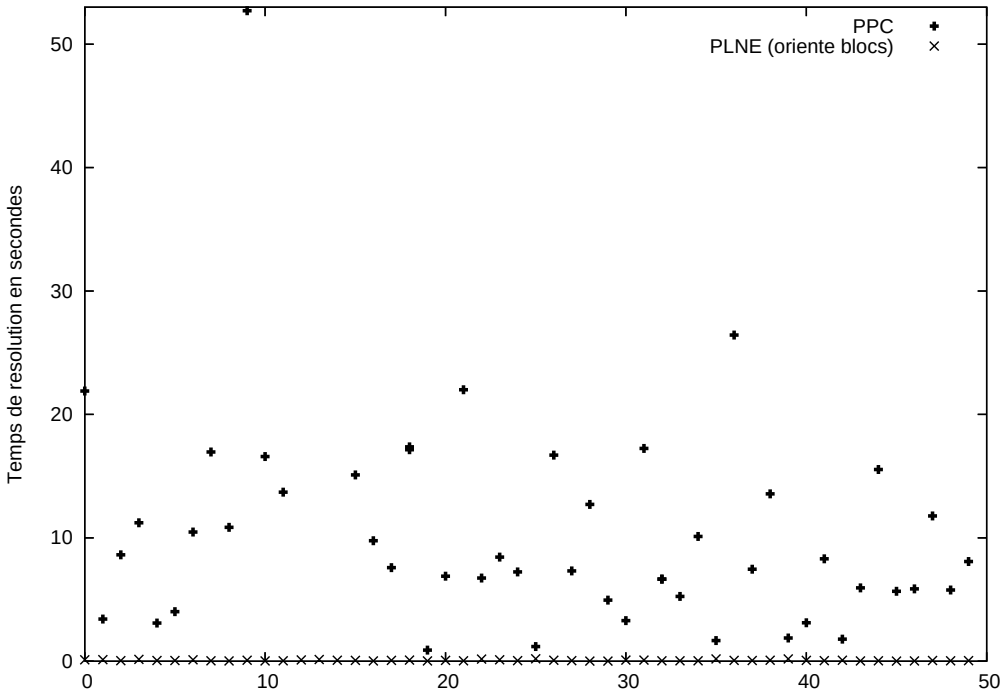


FIG. 6.2 – Temps de calcul des modèles PPC et PLNE pour les instances de F_{15}

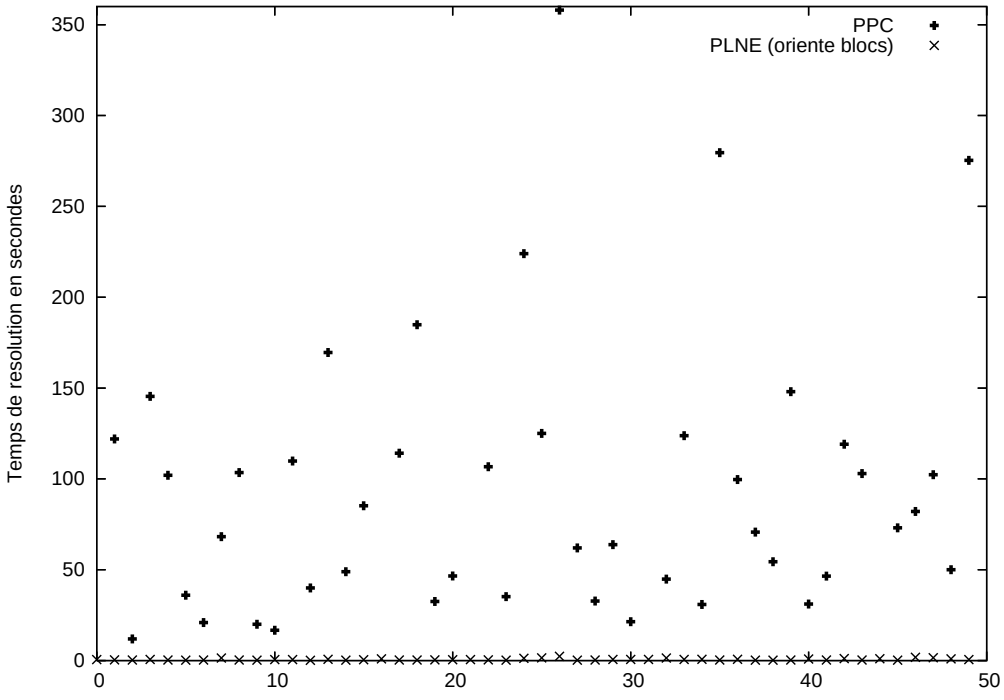


FIG. 6.3 – Temps de calcul des modèles PPC et PLNE pour les instances de F_{20}

des indices de stations pour chaque bloc $[head_b, tail_b]$. Nous avons implémenté l'Algorithme 1 et l'algorithme 2 (voir section 5.4).

À partir de cette section, nous étudions uniquement les familles dont le nombre d'opérations est supérieur à 20, car les autres familles ont des temps de calcul beaucoup trop faible. Pour chaque famille x, nous générons 3 sous-familles $F_x^y, y = 1, 2, 3$ de 50 instances chacune. Les sous-familles appartenant à une même famille se distinguent par la densité du graphe de précédence. Ceci est dû au fait que les algorithmes de réduction exploitent la structure du graphe de précédence, il est donc intéressant de voir l'évolution du temps de calcul en fonction de ce paramètre. Les autres paramètres des sous-familles appartenant à la même famille sont identiques. Nous ne fixons pas la densité exacte mais un intervalle dans lequel elle varie. Ceci est dû à la génération aléatoire des exemples. Nous avons défini trois intervalles (noté par l'indice $y = 1, 2, 3$) pour chaque famille F_x. Les sous-familles F_x^1 ont une faible densité (entre 8 % et 32 %), les F_x^2 ont une densité moyenne (entre 40 % et 65 %) et les F_x^3 une forte densité (entre 60 % et 89 %) du graphe de précédence.

Dans le tableau 6.3, nous montrons le gain apporté. Les colonnes $NbVars1$ et $NbVars2$ contiennent le nombre moyen de variables binaires de la sous-famille correspondante après avoir appliqué l'Algorithme 1 et Algorithme 2, respectivement. Nous indiquons également, dans les colonnes $gain1$ et $gain2$ les pourcentages de gain réalisé en appliquant l'Algorithme 1 et Algorithme 2, respectivement. Ce gain est calculé en utilisant (6.2), il montre l'écart par rapport au nombre initial de variables (soit $m_0 \times Avg|\mathbf{B}|$).

$$gain = \frac{(m_0 \times Avg|\mathbf{B}|) - NbVars}{m_0 \times Avg|\mathbf{B}|} \tag{6.2}$$

Le tableau 6.3 permet de constater que la réduction du nombre de variables binaires en utilisant le calcul des rangs (Algorithme 1) est significative. En observant les différentes sous-familles F_x^y, $x = |\mathbf{N}| = \{20, 30, 40, 50, 60, 70, 80\}$ et $y = 1, 2, 3$, nous apercevons que le gain en nombre de variables est proportionnel à la densité du graphe de précédence. En fait, plus les contraintes de précédence sont nombreuses, plus les rangs des blocs sont serrés et plus la réduction qui les exploitent est efficace. Ce gain peut aller jusqu'à 72 % pour la sous-famille F_{60}^3 dont la densité est dans l'intervalle $[80, 89]$.

L'Algorithme 1 donne déjà un gain intéressant. L'Algorithme 2 permet d'en réaliser davantage. Ce dernier apporte un gain supplémentaire d'au moins[2] 2% par rapport à l'Algorithme 1. Ce gain peut même atteindre 10 % dans certains cas, par exemple pour les familles F_{50}^1 et F_{70}^2.

6.3.2 Temps de calcul

La figure 6.4 représente les temps d'exécution des instances appartenant aux sous-familles de 20 et 30 opérations. Pour chaque sous-famille F_x^y nous donnons trois résultats, le premier, noté A_0, est le temps de résolution sans l'application d'algorithmes de réduction. Le second,

[2]Par rapport à l'échantillon de familles testées.

					Algo1		**Algo2**	
	$\|\mathbf{N}\|$	OSI	OS_{avg}	$m_0 \times Avg\|\mathbf{B}\|$	$NbVars1$	gain1	$NbVars2$	gain2
F_{20}^1		$[8, 15]$	12		330	**15** %	324	**17** %
F_{20}^2	20	$[40, 50]$	35	13×30 = 390	263	**32** %	253	**35** %
F_{20}^3		$[60, 81]$	63		183	**53** %	175	**55** %
F_{30}^1		$[15, 32]$	22		565	**26** %	547	**29** %
F_{30}^2	30	$[42, 65]$	52	16×48 = 768	416	**46** %	394	**49**%
F_{30}^3		$[62, 80]$	70		287	**63** %	262	**66**%
F_{40}^1		$[15, 29]$	20		1114	**22** %	1084	**24**%
F_{40}^2	40	$[40, 60]$	52	21×68 = 1428	817	**42** %	788	**45**%
F_{40}^3		$[68, 88]$	79		440	**69** %	400	**72**%
F_{50}^1		$[17, 31]$	24		1885	**22** %	1826	**32**%
F_{50}^2	50	$[47, 64]$	54	26×93 = 2418	1442	**40** %	1385	**43**%
F_{50}^3		$[78, 87]$	83		773	**68** %	646	**73**%
F_{60}^1		$[9, 18]$	13		3019	**11** %	2944	**13**%
F_{60}^2	60	$[47, 64]$	58	30×113 = 3390	2133	**37** %	2008	**41**%
F_{60}^3		$[80, 89]$	84		951	**72** %	854	**75**%
F_{70}^1		$[12, 19]$	15		3817	**16** %	3708	**18,5**%
F_{70}^2	70	$[44, 62]$	51	34×133 = 4522	3000	**34** %	2854	**44**%
F_{70}^3		$[79, 88]$	84		1428	**68** %	1255	**75**%
F_{80}^1		$[12, 28]$	18		4879	**16** %	4751	**19**%
F_{80}^2	80	$[43, 56]$	49	38×152 = 5776	4066	**30** %	3920	**33**%
F_{80}^3		$[80, 89]$	84		1918	**66** %	1712	**70**%

TAB. 6.3 – Analyse comparative des algorithmes de réduction du nombre de variables

libellé A_1, indique les temps de résolution des instances après le pré-traitement avec l'Algorithme 1. Enfin, le troisième résultat, nommé A_2, correspond à l'utilisation de l'Algorithme 2. Ainsi, pour comparer les performances des algorithmes pour une sous-famille donnée F_x^y, $x = 20, 30$ et $y = 1, 2, 3$, il faut observer les trois abscisses F_x^y : A_0, A_1 et A_2. Il est important de préciser que nous employons une échelle logarithmique, car la variation des résultats est importante.

Sur l'ensemble de ces sous-familles, nous constatons un gain considérable du temps de résolution suite à l'application de l'Algorithme 1. En particulier, pour les sous-familles de 30 opérations, l'ensemble des indicateurs notamment la médiane est divisée par 10 grâce à l'Algorithme 1. Ce qui signifie que plus de 50 % des instances sont résolues 10 fois plus rapidement. L'application de l'Algorithme 2 permet davantage d'accélération dans les calculs. Pour la famille F_{30}, le gain obtenu avec l'Algorithme 2 par rapport à celui de l'Algorithme 1 n'est toutefois pas aussi important que le gain réalisé par l'Algorithme 1 par rapport à A_0 (sans l'utilisation d'algorithmes de réduction) pour la même famille.

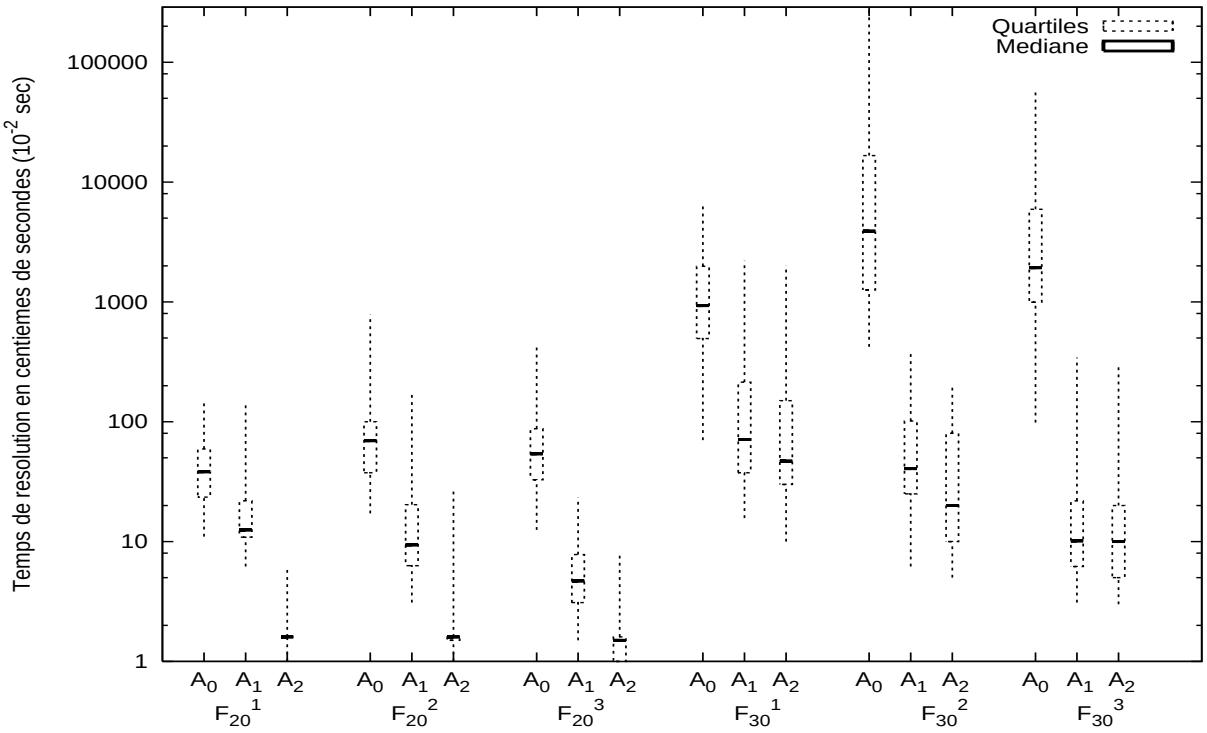


FIG. 6.4 – Le gain obtenu avec les Algorithmes 1 et 2 pour les instances de F_{20}^x et F_{30}^x

6.3.3 Apport de la modification de l'objectif (epsilon)

Pour cette partie des expérimentations, nous appliquons l'Algorithme 1 et testons l'impact de la modification de l'objectif en introduisant une constante ε dans la formulation de l'objectif. Nous résolvons les familles de 20, 30, 40, 50 et 60 opérations avec les deux formulations de l'objectif, à savoir : avant et après l'introduction de l'*epsilon* (voir respectivement, (5.17) et (5.35)).

Les abscisses dans la figure 6.5 sont à observer par paires successives. Nous utilisons un asterisk pour désigner les temps d'exécution après modification de l'objectif en introduisant l'*epsilon*.

Pour les familles de 20 et 30 opérations, soit la figure 6.5, la modification suite à l'ajout d'un epsilon n'améliore pas systématiquement les temps de calcul. En effet, nous remarquons une amélioration des temps pour trois sous-familles : F_{20}^1, F_{20}^2 et F_{30}^1, tandis que nous notons une détérioration des 3 autres sous-familles. Nous pensons que l'emploi de l'epsilon pour ces instances de petites taille a dû perturber l'ordre de branchements ce qui a eu pour effet de ralentir le processus de résolution.

Pour les instances de taille supérieure, c'est-à-dire celles appartenant aux familles de 40, 50 et 60 opérations, nous constatons une nette amélioration (voir les figures 6.6 et 6.7). Le gain apporté est significatif car il permet une réduction de 50% du temps de résolution par rapport au modèle initial utilisant la formulation (5.17).

Dans la figure 6.8, nous rapportons les temps d'execution de toutes les instances des

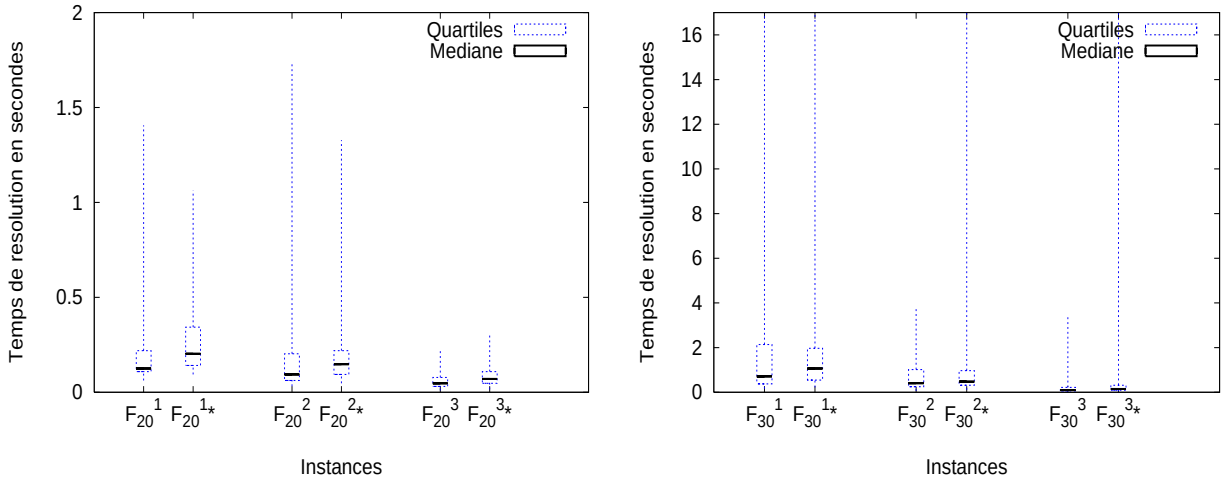


FIG. 6.5 – Influence de la modification de l'objectif pour les instances de 20 et 30 opérations

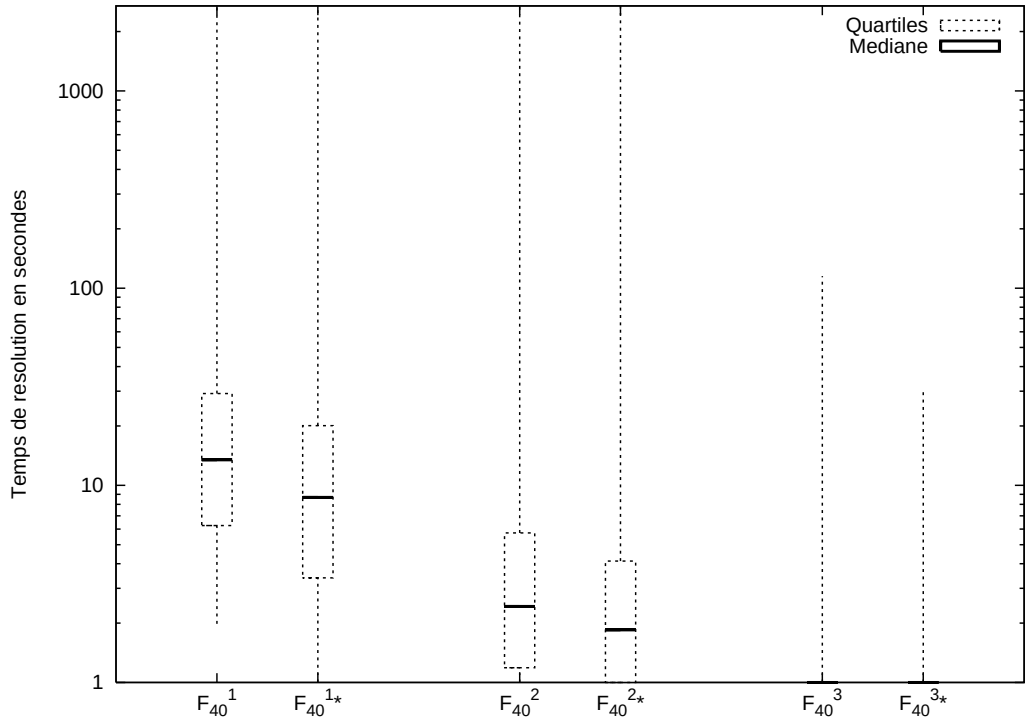


FIG. 6.6 – Influence de la modification de l'objectif pour les familles de 40 opérations

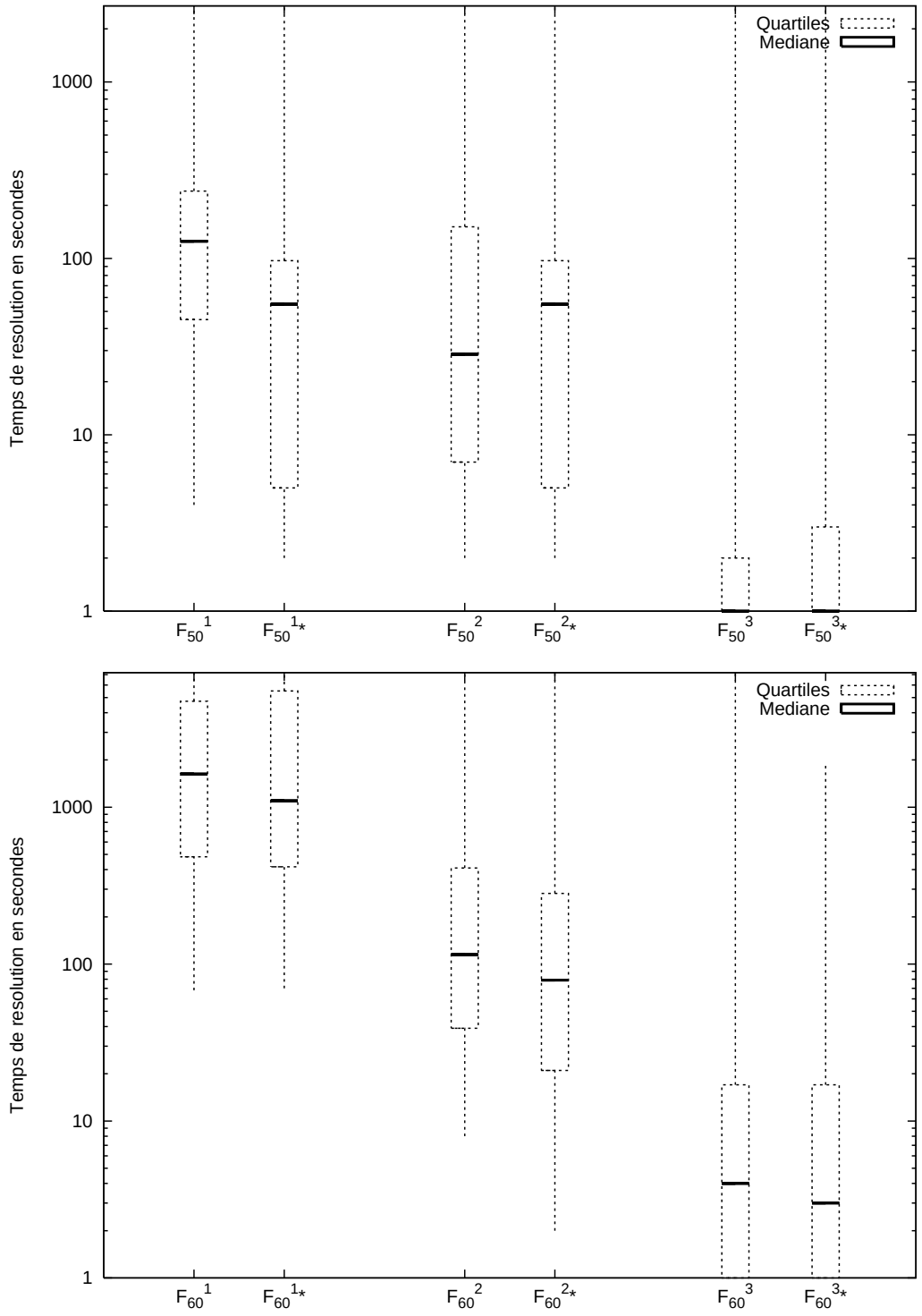


FIG. 6.7 – Influence de la modification de l'objectif pour les familles de 50 et 60 opérations

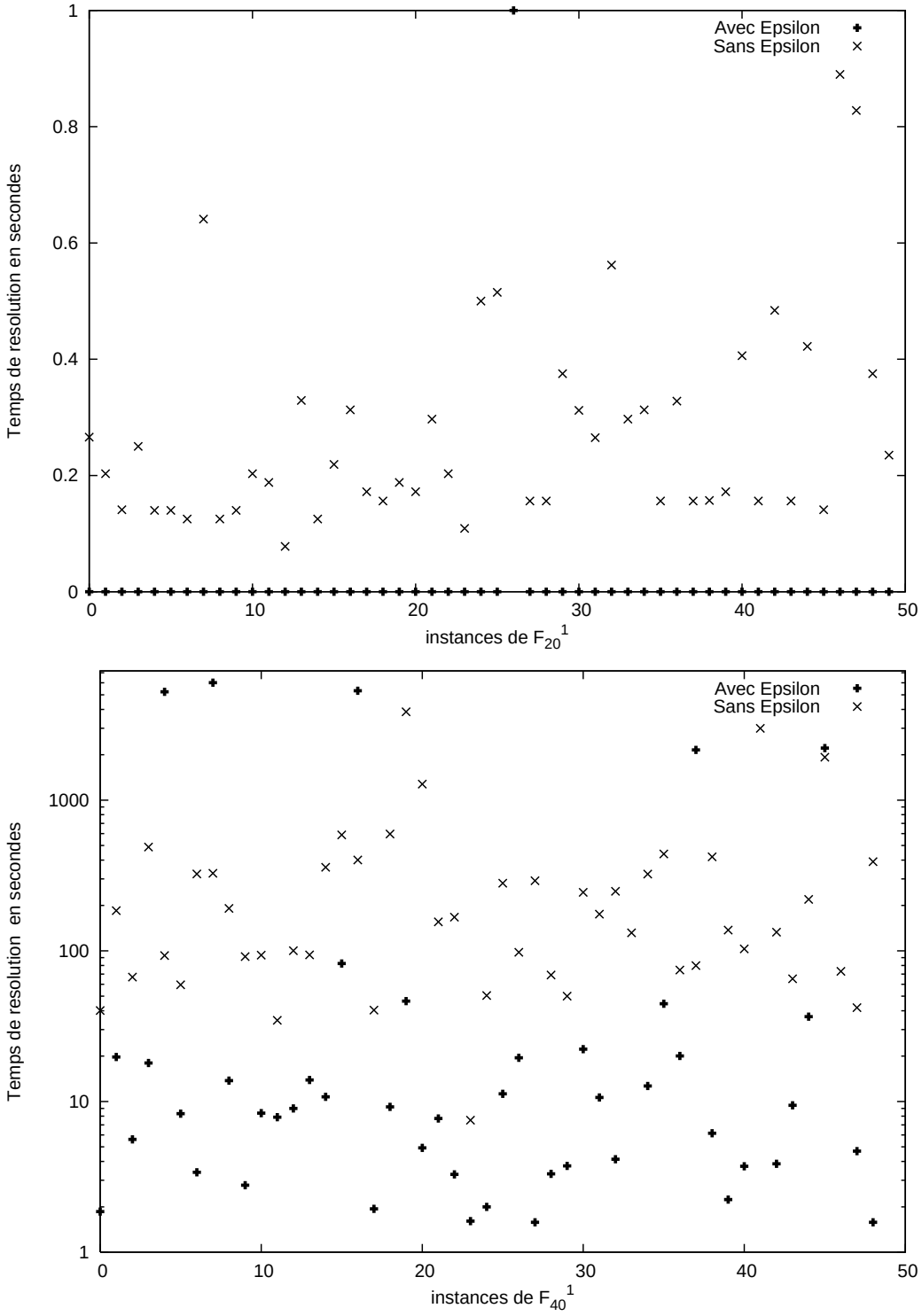


FIG. 6.8 – Influence de la modification de l'objectif pour F_{20}^1 et F_{40}^1

sous-familles F_{20}^1 et F_{40}^1 telle que chaque abscisse représente une instance (nuage de points). Ces résultats permettent de voir l'amélioration apportée par la modification de l'objectif pour chacune des instances.

Pour conclure, nous pouvons dire que l'amélioration apportée par la reformulation de l'objectif en introduisant une constante ε est significative pour les cas les plus difficiles à résoudre, toutefois cette réduction n'est pas garantie pour toute instance. En effet, concernant les instances de petite taille cette modification peut engendrer une perturbation dans le schéma de résolution de Cplex et donc le ralentir, aussi il n'est pas forcément judicieux de l'utiliser, pour de telles instances, d'autant plus que le temps de calcul n'est pas très important dans ces cas (n'excède pas les 20 secondes).

6.4 Modèle orienté opérations

Dans cette partie des expérimentations, l'Algorithme 2 est employé dans la phase de réduction préliminaire. Afin que la comparaison avec le modèle orienté blocs soit valide, nous avons également appliqué l'Algorithme 2 pour le modèle orienté blocs.

6.4.1 Comparaison avec le modèle orienté blocs

Nous rapportons les résultats des tests pour les familles de 20, 30, 40, 50 et 60 opérations que nous avons résolues avec les deux modèles linéaires. Les abscisses libellées MB indiquent les résultats obtenus en utilisant le modèle orienté blocs alors que celles libellées MO indiquent ceux du modèle orienté opérations.

Dans la figure 6.9, nous montrons les temps de résolution de la famille F_{20} alors que dans la figure 6.10 nous rapportons les temps de chacune des instances appartenant à la sous-famille F_{20}^1. Les figures 6.11 et 6.12 montrent les résultats obtenus pour les famille de 30 à 60 opérations. Pour les familles 40, 50 et 60 opérations, l'échelle des graphiques est logarithmique. Par exemple, le temps maximum pour la sous-famille F_{40}^1 est égale à 7200 sec pour le modèle orienté blocs alors qu'il est ramené à 11 sec dans le modèle orienté opérations.

En observant l'ensemble des figures, nous constatons une forte réduction des temps d'exécution en faveur du modèle orienté opérations. En effet, le modèle orienté opérations permet une résolution en un dixième de seconde pour les instances appartenant aux familles de 20 et de 30 opérations (à l'exception d'une instance de la sous-famille F_{30}^1 qui est résolue en 2.5 sec, voir le maximum $F_{30}^1 : MO$).

Concernant les familles de 40, 50 et 60 opérations, la limite de temps de 7200 sec accordée aux deux modèles est systématiquement dépassée pour le modèle orienté blocs (sauf pour F_{60}^3). Ainsi, pour les instances difficiles soit celles de F_x^1, où $x \in \{40, 50, 60\}$), le modèle orienté blocs est incapable d'apporter les preuves de l'optimalité des solutions trouvées tandis que le modèle orienté opérations résout à l'optimum l'ensemble des instances en réduisant le temps de calcul de façon considérable (pour F_{60}^1 le temps de calcul maximum dépasse la limite 7200 avec le modèle orienté blocs il est réduit à 32 sec à l'aide du modèle orienté opérations).

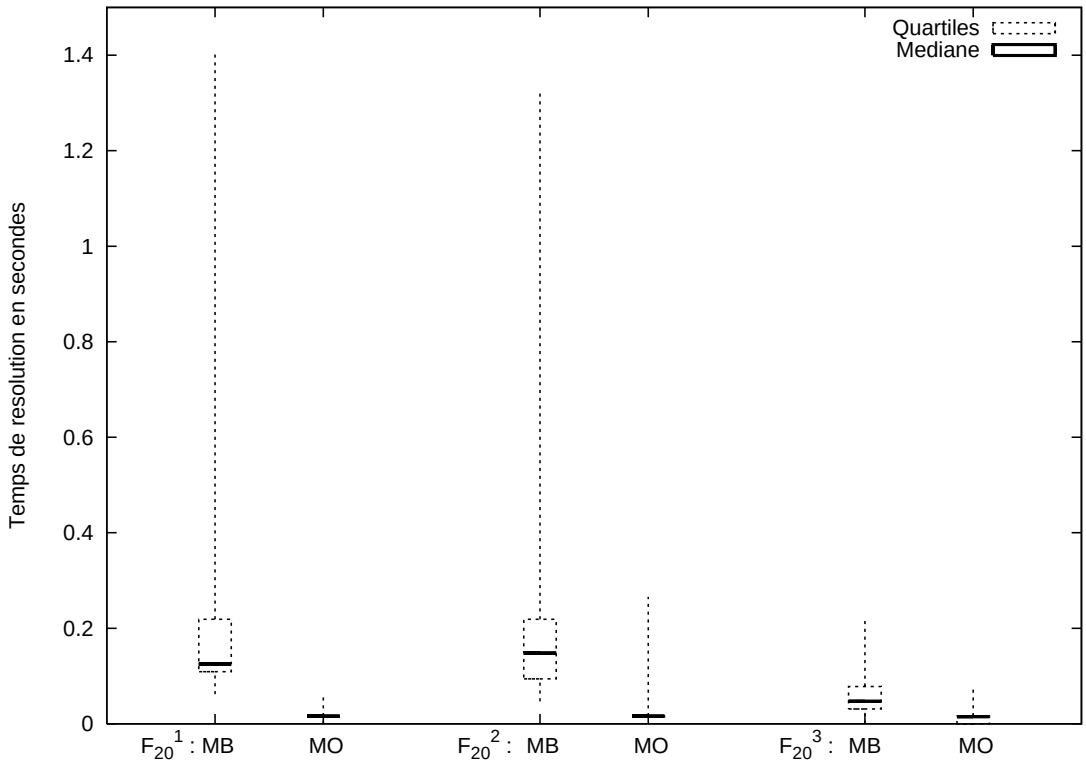


FIG. 6.9 – Résolution des familles de 20 opérations avec les deux modèles linéaires

Pour conclure, nous dirons que le modèle orienté opérations est plus performant tant sur le plan du temps de calcul des instances difficiles que pour sa stabilité (résolvant des instances jusqu'à 80 opérations).

6.4.2 Étude de l'influence des précédences (OS)

La figure 6.13 permet de voir l'évolution des temps de résolution pour les familles de 20, 30, 40 et 50 opérations. Pour une même famille, il faut observer les trois abscisses libellées F_x^1, F_x^2 et F_x^3, elles correspondent aux sous-familles avec faible, moyenne et forte densité, respectivement. Nous constatons une réduction significative des temps de calcul pour les sous-familles ayant une plus grande densité du graphe de précédence. La même conclusion peut être faite pour les familles de 60, 70 et 80 opérations dont les résultats sont dans la figure 6.14. Les médianes des sous-familles F_x^3 par rapport à celles des F_x^1, sont divisées par 10 ou 100 selon les familles (100 fois pour les 60 et 70 et plus pour 80 opérations).

De plus, une influence directe du nombre d'opérations sur le temps d'execution est à noter. Les sous-familles de même densité semblent requérir un temps de calcul croissant en fonction du nombre d'opérations. La sous-famille de F_{80}^1 est la plus difficile à résoudre par rapport aux autres sous-familles de type F_x^1. En effet, pour cette sous-famille, plus de 25% des instances (voir le 3$^{\text{ème}}$ quartile de F_{80}^1 dans la figure 6.14) nécessitent un temps de résolution

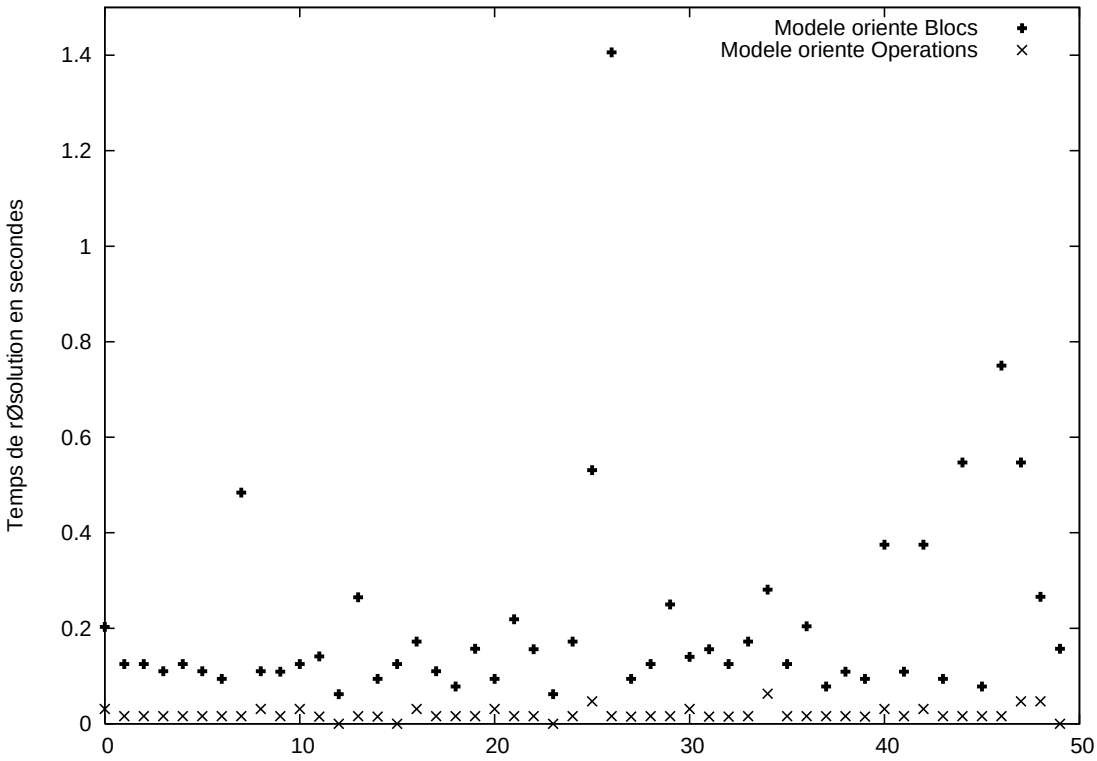


FIG. 6.10 – Temps de calcul de F_{20}^1 avec les deux modèles linéaires

supérieur à la limite que nous avons accordé au solver, soit 7200 sec. De la même manière, les sous-familles F_{80}^2 et F_{80}^3 sont plus coûteuse en temps de calcul que F_x^2, F_x^3, respectivement tel que $x < 80$.

Sur la base des expérimentations comprenant 1050 tests, nous constatons également un lien direct entre la densité du graphe de précédence et le temps de calcul des instances. En observant chacune des familles séparément, nous constatons une nette diminution du temps d'exécution lorsque la densité OS augmente. Ainsi, pour chaque famille, la sous-famille à faible densité est toujours plus difficile à résoudre que celle ayant une densité moyenne. De même, les sous-familles ayant une moyenne densité sont plus coûteuses en temps de calcul que celles ayant une forte densité.

La difficulté des instances à faible densité peut s'expliquer par le grand nombre d'affectations possibles des blocs aux stations. Ceci a une répercussion directe sur le nombre de branchements à générer lors de la construction de l'arbre de séparation et d'évolution par Cplex. Toutefois, ce paramètre n'est pas le seul à influencer le temps de calcul d'autant plus que la difficulté de certaines instances peut être due à une combinaison particulière de données.

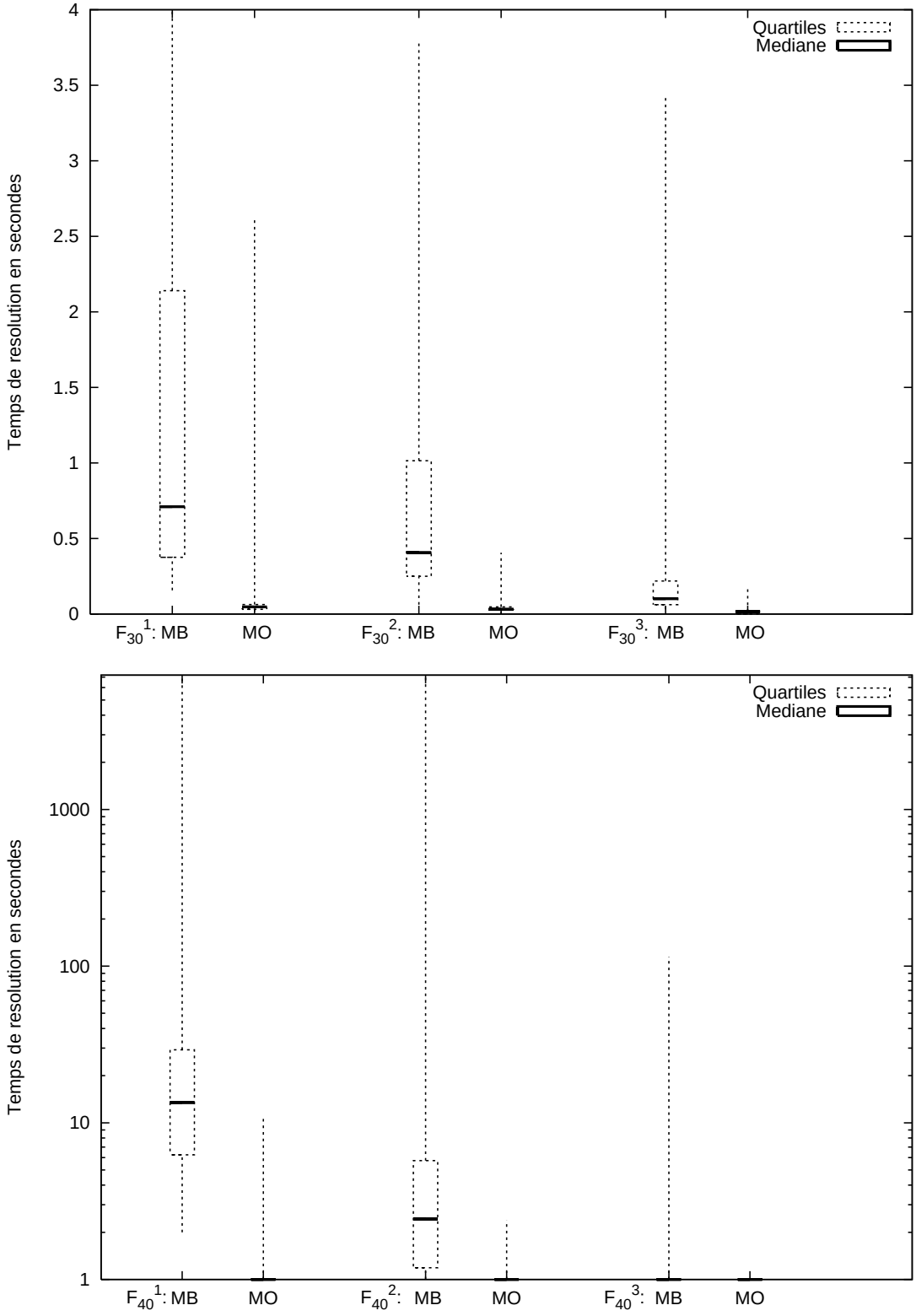


FIG. 6.11 – Résolution des familles de 30 et 40 opérations avec les deux modèles linéaires

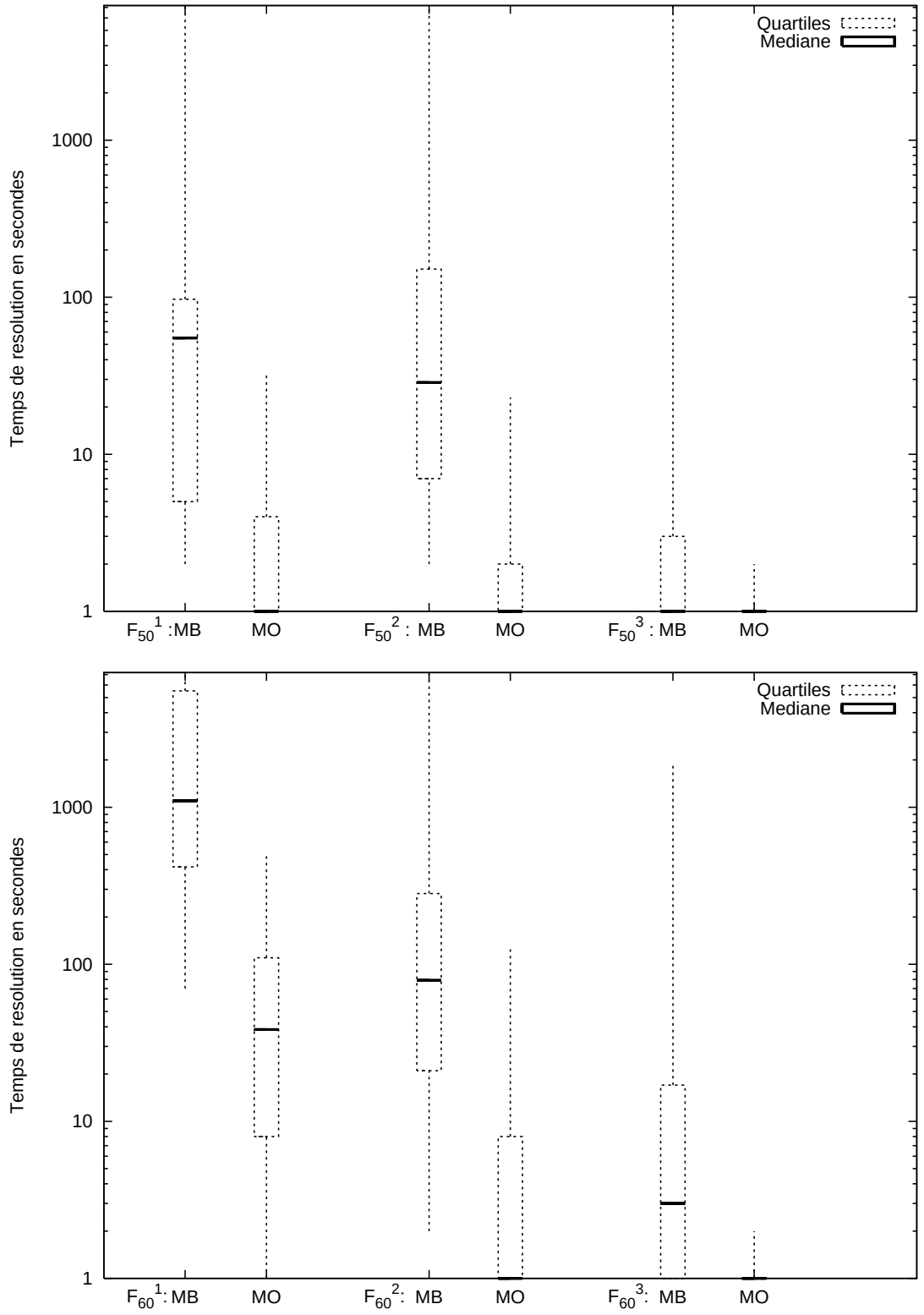


FIG. 6.12 – Résolution des familles de 50 et 60 opérations avec les deux modèles linéaires

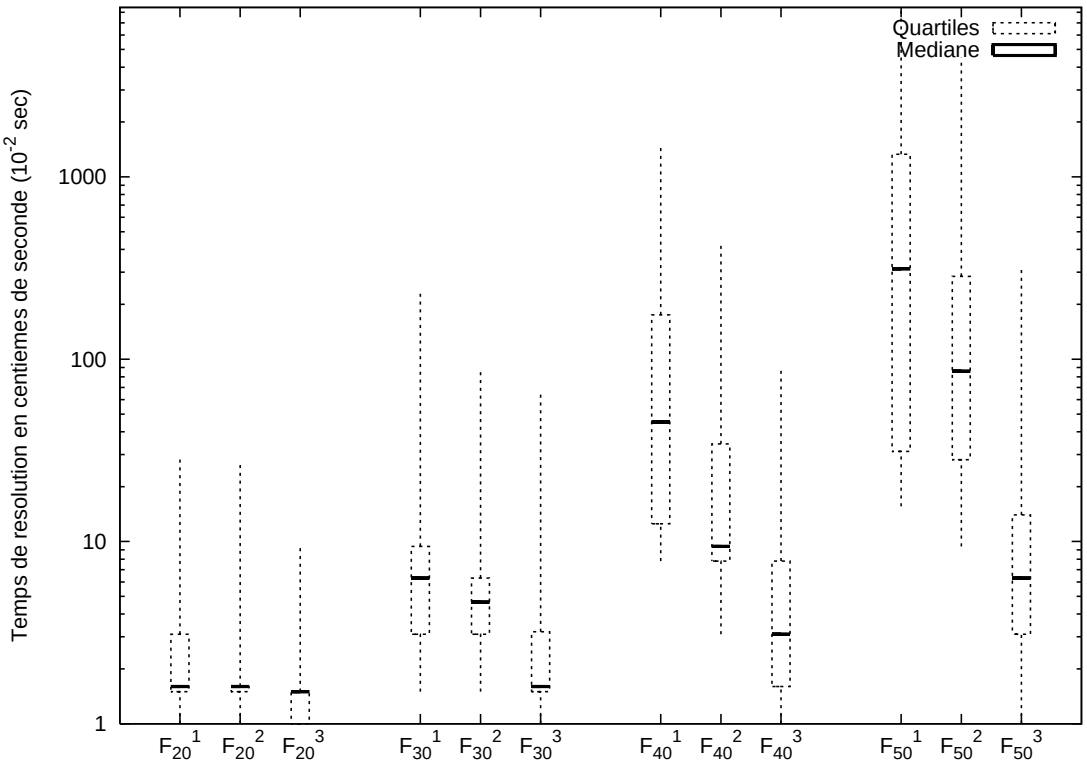


FIG. 6.13 – Temps de calcul en fonction de OS pour les familles de 20, 30, 40 et 50 opérations

6.4.3 Étude de l'influence du nombre de blocs (|B|)

Nous consacrons la présente sous-section pour l'étude de l'influence de la cardinalité de l'ensemble **B** sur le temps de résolution. Le tableau 6.4 décrit les familles d'instances qui ont été employées dans cette partie des expérimentations. Pour chaque famille, nous avons généré trois sous-familles comportant 50 instances chacune. Les sous-familles ont les mêmes caractéristiques : $|\mathbf{N}|$, m_0 et OSI.

Les sous-familles notées $F_x^{B_1}$ ont un nombre moyen de blocs égale à une fois et demi le nombre d'opérations. Par exemple, la sous-famille $F_{20}^{B_1}$ à un nombre moyen de blocs $Avg|\mathbf{B}|$= 30 = 1,5×20. Les sous-familles $F_x^{B_2}$ ont un nombre moyen de blocs qui est égal au double du nombre d'opérations. Quant aux dernières sous-familles : $F_x^{B_3}$, elles ont un nombre de blocs, égal au triple du nombre d'opérations.

La figure 6.15 montre le temps d'exécution pour les familles de 20, 30, 40, 50, 60 et 70 opérations. Nous constatons que les sous-familles $F_x^{B_3}$ sont plus coûteuses en temps de calcul par rapport aux $F_x^{B_1}$ et $F_x^{B_2}$. Leur temps de résolution augmente de façon significative : la médiane de $F_{20}^{B_3}$ est deux fois plus élevée que celle de $F_{20}^{B_1}$. Cette différence est encore plus visible pour la famille de 30 et 40 opérations, où la médiane de $F_{30}^{B_3}$ est 7 fois plus élevée que celle de $F_{30}^{B_2}$ et celle de $F_{40}^{B_3}$ est 15 fois plus importante que la médiane de $F_{40}^{B_2}$.

Le même constat peut être fait en comparant $F_x^{B_2}$ avec $F_x^{B_1}$ (à l'exception de $F_{20}^{B_2}$ et $F_{20}^{B_1}$

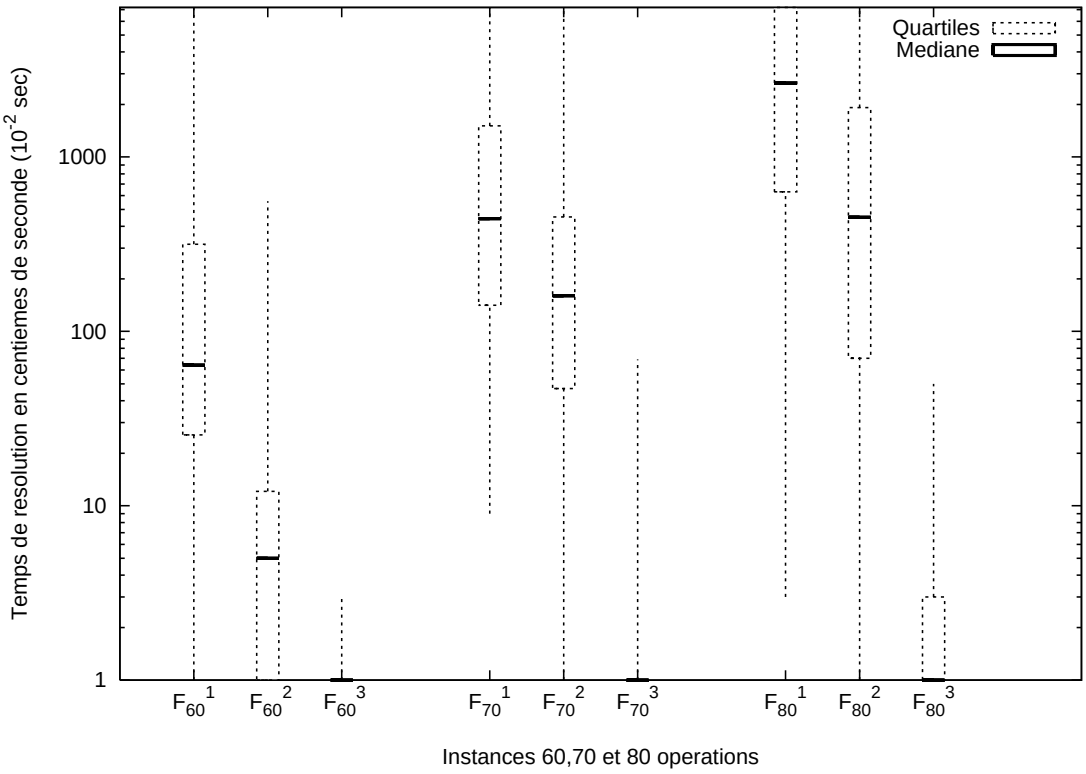


FIG. 6.14 – Temps de calcul en fonction OS pour les familles de 60, 70 et 80 opérations

car dans les deux cas la résolution des instances ne dépassent pas une seconde).

Nous observons le même phénomène pour les familles de 50, 60 et 70 opérations. Par ailleurs, nous constatons que les limites du modèle sont atteintes pour la famille $F_{70}^{B_3}$ dont le nombre moyen de blocs disponibles est de 210. La médiane de la sous-famille $F_{70}^{B_3}$ est égale au temps alloué à Cplex, ce qui signifie qu'il y a au moins 50 % des instances pour lesquelles il n'a pas été fourni une preuve de l'optimalité (les meilleures solutions trouvées peuvent même être juste réalisables). Nous observons un autre cas de dépassement du temps alloué lorsque le nombre de blocs est égal à 180 et le nombre d'opérations est à 60 ($F_{60}^{B_3}$). Dans ce cas, plus d'un quart des instances de la sous-famille $F_{60}^{B_3}$ n'ont pu être résolues à l'optimum, car le $3^{\text{ème}}$ quartile coincide avec le maximum de temps accordé (7200 sec).

6.4.4 Étude de l'influence du nombre d'opérations par bloc

Cette sous-section est réservée à l'étude de l'impact du nombre moyen d'opérations par bloc sur le temps de calcul. Afin de réaliser cette étude nous avons généré une autre série d'instances. Pour chaque famille F_x tel que $x \in \{20, 30, 40, 50, 60, 70\}$ nous avons généré trois sous-familles, notée comme suit :

$F_x^{O_1}$, $F_x^{O_2}$ et $F_x^{O_3}$. Les sous-familles de type $F_x^{O_1}$ ont l'intervalle $OpI = [OpMin, OpMax]$

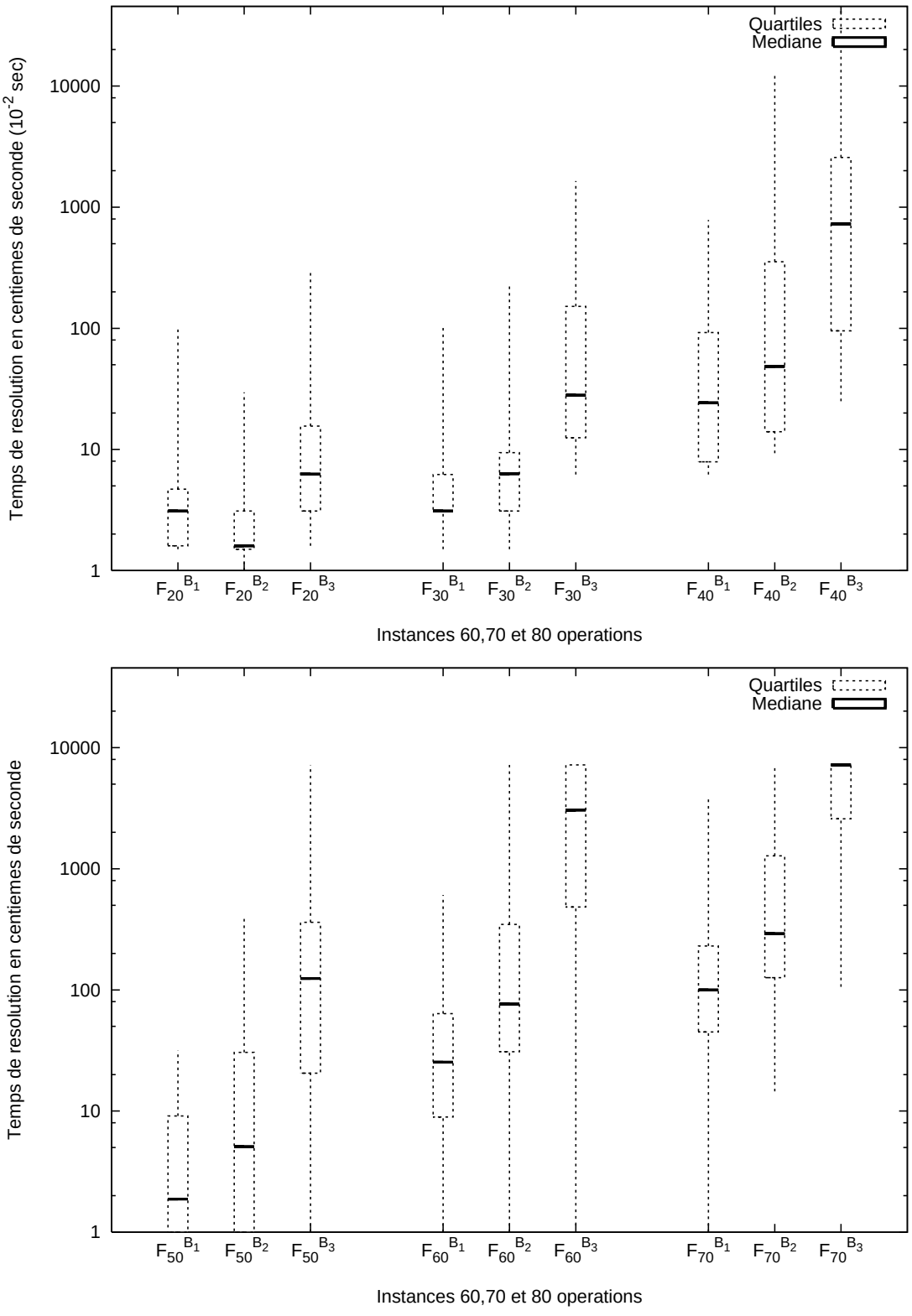


FIG. 6.15 – Temps de calcul pour l'étude de l'influence de $|\mathbf{B}|$

$F_x^{B_i}$	$\|\mathbf{N}\|$	$\|\mathbf{B}\|$	OSI	OS_{avg}	m_0	$F_x^{B_i}$	$\|\mathbf{N}\|$	$\|\mathbf{B}\|$	OSI	OS_{avg}	m_0
$F_{20}^{B_1}$		30				$F_{50}^{B_1}$		75			
$F_{20}^{B_2}$	20	40	$[8, 15]$	12	13	$F_{50}^{B_2}$	50	100	$[17, 31]$	24	26
$F_{20}^{B_3}$		60				$F_{50}^{B_3}$		150			
$F_{30}^{B_1}$		45				$F_{60}^{B_1}$		90			
$F_{30}^{B_2}$	30	60	$[15, 32]$	22	16	$F_{60}^{B_2}$	60	120	$[9, 18]$	13	30
$F_{30}^{B_3}$		90				$F_{60}^{B_3}$		180			
$F_{40}^{B_1}$		60				$F_{70}^{B_1}$		105			
$F_{40}^{B_2}$	40	80	$[15, 29]$	20	21	$F_{70}^{B_2}$	70	140	$[12, 19]$	15	34
$F_{40}^{B_3}$		120				$F_{70}^{B_3}$		210			

TAB. 6.4 – Description des instances pour étudier l'influence de $|\mathbf{B}|$

égal à [1,2], ce qui correspond aux instances dont les blocs contiennent une ou deux opérations. Les sous-familles de type $F_x^{O_2}$ correspondent à l'intervalle OpI =[2,5]. Les sous-familles $F_x^{O_3}$ sont les instances dans lesquelles les blocs peuvent contenir de 3 à 7 opérations, l'intervalle $OpI = [3, 7]$. Nous rapportons dans le tableau 6.5 l'ensemble des caractéristiques des instances qui ont été générées.

$F_x^{O_i}$	$\|\mathbf{N}\|$	OpI	$Avg\|\mathbf{B}\|$	m_0	$F_x^{O_i}$	$\|\mathbf{N}\|$	OpI	$Avg\|\mathbf{B}\|$	m_0
$F_{20}^{O_1}$		[1,2]			$F_{50}^{O_1}$		[1,2]		
$F_{20}^{O_2}$	20	[2,5]	32	13	$F_{50}^{O_2}$	50	[2,5]	95	26
$F_{20}^{O_3}$		[3,7]			$F_{50}^{O_3}$		[3,7]		
$F_{30}^{O_1}$		[1,2]			$F_{60}^{O_1}$		[1,2]		
$F_{30}^{O_2}$	30	[2,5]	45	16	$F_{60}^{O_2}$	60	[2,5]	110	30
$F_{30}^{O_3}$		[3,7]			$F_{60}^{O_3}$		[3,7]		
$F_{40}^{O_1}$		[1,2]			$F_{70}^{O_1}$		[1,2]		
$F_{40}^{O_2}$	40	[2,5]	60	21	$F_{70}^{O_2}$	70	[2,5]	130	34
$F_{40}^{O_3}$		[3,7]			$F_{70}^{O_3}$		[3,7]		

TAB. 6.5 – Description des instances pour l'étude de l'influence de OpI

La figure 6.16 montre les résultats de la résolution de ces instances. Les sous-familles de 20 opérations sont du même ordre de difficulté, la plupart des instances de cette famille sont solvables en moins de 3 centièmes de seconde. Par contre, une différence significative pour les familles de 30 et 40 opérations est observée. En effet, pour chacune de ces familles, nous observons une nette diminution de la médiane au fur et à mesure que le nombre d'opérations par bloc augmente. Ce résultat s'explique par le fait qu'un nombre élevé d'opérations par bloc engendre des solutions réalisables comportant moins de blocs. Ainsi, lors de la construction

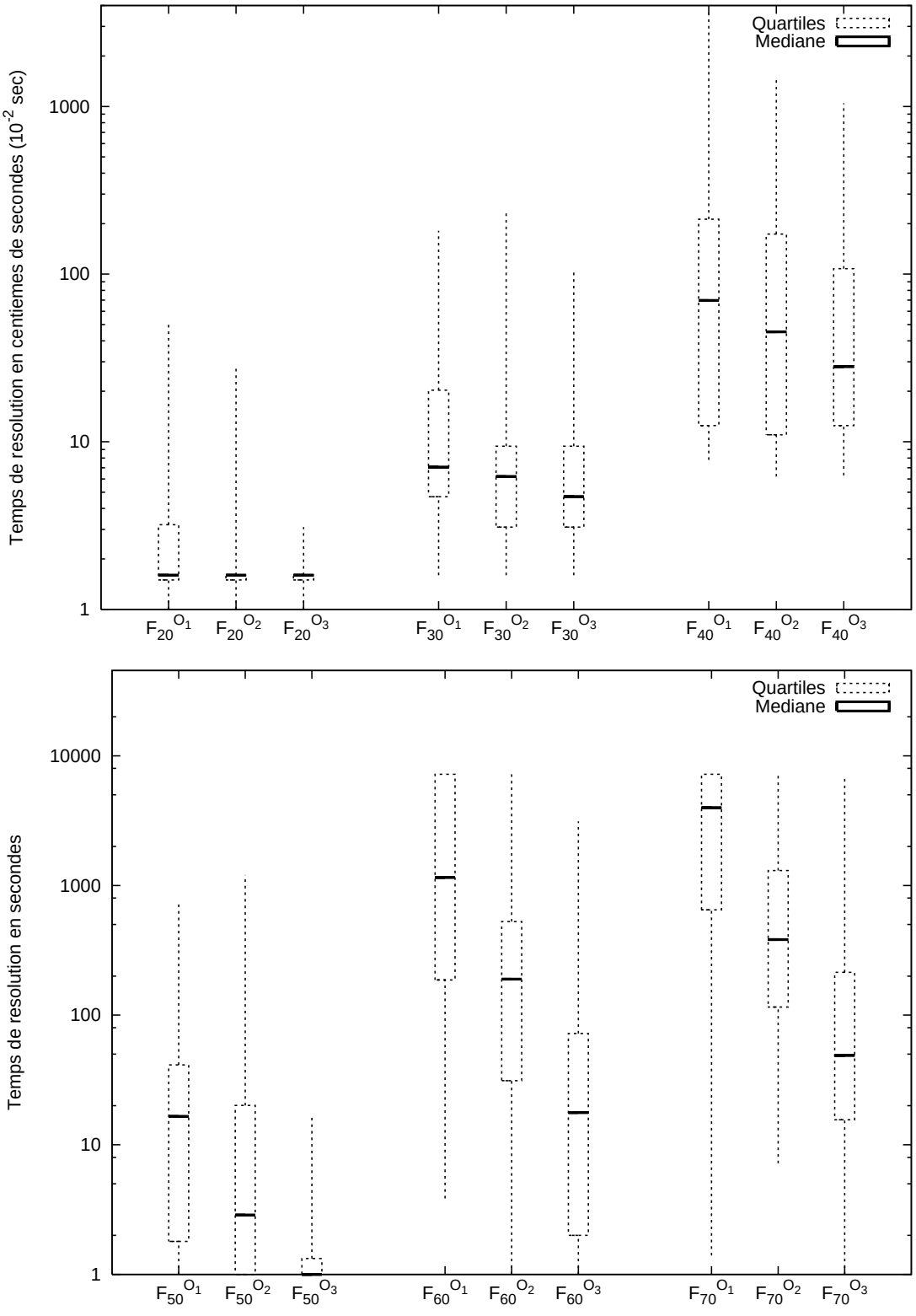


FIG. 6.16 – Temps de calcul en fonction de OpI

de l'arbre d'énumération dans la PSE utilisée Cplex les feuilles sont atteintes plus rapidement ce qui a pour effet de réduire le nombre total de branchements.

Il arrive toutefois qu'une instance ayant un nombre d'opérations par bloc élevé soit plus difficile à résoudre qu'une autre instance ayant les mêmes valeurs pour les autres paramètres mais avec un nombre d'opérations par blocs moins grand. Cette situation est constatée pour les instances les plus difficiles des sous-familles $F_{30}^{O_1}$ et $F_{30}^{O_2}$ (voir les points extrêmes hauts des candlesticks de $F_{30}^{O_1}$ et $F_{30}^{O_2}$). En effet, un jeu de données d'une instance peuvent la rendre difficile à résoudre et ce malgré son appartenance à une sous-famille, *a priori*, facile.

6.4.5 Étude de l'effet du nombre maximum de stations

Le nombre de stations maximum m_0 avec le nombre de blocs $|\mathbf{B}|$ déterminent le nombre de variables binaires x_{bk}. Ce paramètre a donc, *a priori*, une influence directe sur les performances des modèles. Afin de s'en assurer, nous avons généré une autre série de tests dans laquelle nous avons fait varier le nombre maximum de stations. Nous rapportons dans le tableau 6.6 les paramètres utilisés pour la génération de ces sous-familles $F_x^{m_i}$, $i = 1, 2, 3$ et $x \in \{20, 30, 40, 50, 60, 70\}$.

$F_x^{m_i}$	$\lvert\mathbf{N}\rvert$	m_0	$Avg\lvert\mathbf{B}\rvert$	OsI	$F_x^{m_i}$	$\lvert\mathbf{N}\rvert$	m_0	$Avg\lvert\mathbf{B}\rvert$	OsI
$F_{20}^{m_1}$		7			$F_{50}^{m_1}$		12		
$F_{20}^{m_2}$	20	10	32	$[8, 15]$	$F_{50}^{m_2}$	50	19	95	$[17, 31]$
$F_{20}^{m_3}$		13			$F_{50}^{m_3}$		26		
$F_{30}^{m_1}$		10			$F_{60}^{m_1}$		14		
$F_{30}^{m_2}$	30	13	45	$[15, 32]$	$F_{60}^{m_2}$	60	20	110	$[9, 18]$
$F_{30}^{m_3}$		16			$F_{60}^{m_3}$		30		
$F_{40}^{m_1}$		12			$F_{70}^{m_1}$		14		
$F_{40}^{m_2}$	40	17	60	$[15, 29]$	$F_{70}^{m_2}$	70	20	130	$[12, 19]$
$F_{40}^{m_3}$		21			$F_{70}^{m_3}$		34		

TAB. 6.6 – Description des instances générées pour l'étude de l'influence de m_0

Nous rapportons dans les figures 6.17 et 6.18 le temps de calcul pour les instances décrites dans le tableau 6.6. Les résultats confirment que ce modèle est plus performant pour les instances appartenant à des sous-familles dont la valeur de m_0 est faible. Les instances de 70 opérations marquent le plus grand écart. En effet, l'instance la plus difficile de $F_{70}^{m_1}$ nécessite à peine 24 sec contre plus de 7200 sec pour celle de $F_{70}^{m_3}$.

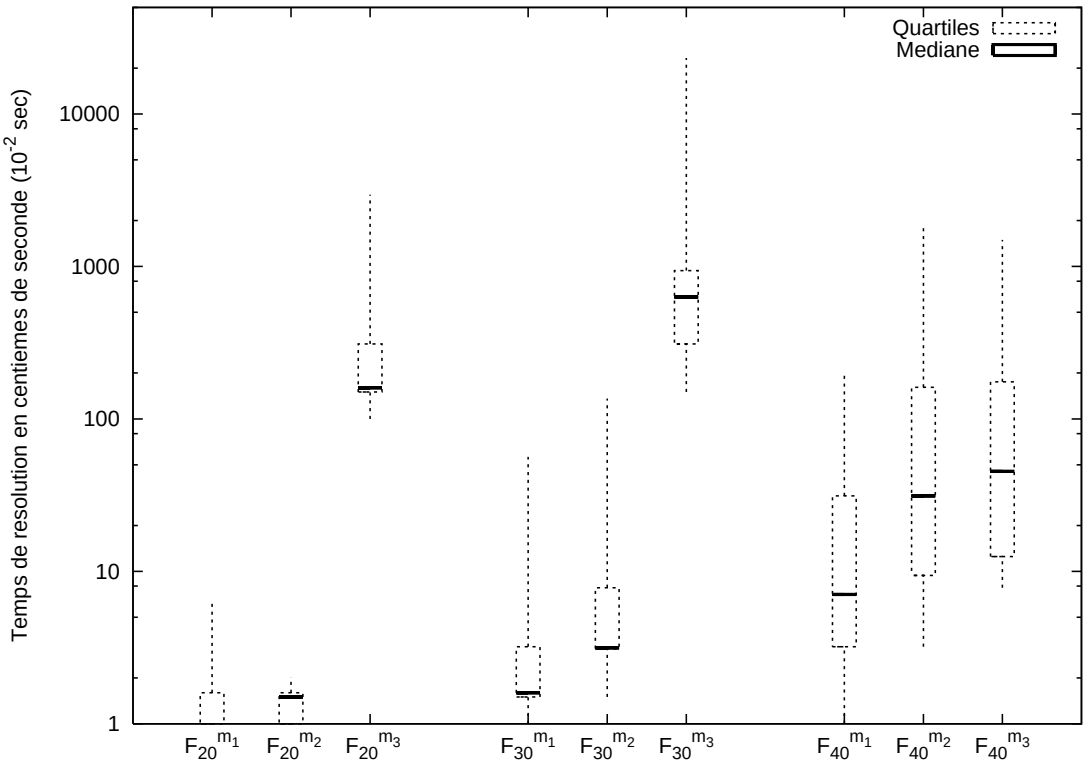


FIG. 6.17 – Influence de m_0 pour les familles de 20, 30 et 40 opérations

6.5 Comparaison avec une approche graphe

Dans cette sous-section, nous présentons l'étude comparative des performances entre le modèle orienté opérations et l'approche graphe qui a été proposée dans [DGL06] (voir section 3.6.1).

Pour cette étude, nous utilisons les familles d'instances décrites dans la sous-section 6.4.2. Nous rapportons les résultats obtenus dans les figures 6.19. Nous indiquons, pour chaque sous-famille F_x^y, le temps de calcul de la méthode graphe par *GRP* et celui du modèle linéaire orienté opérations par *MO*.

Ces figures permettent de constater une nette domination du modèle linéaire pour toutes les familles sauf pour les instances les plus difficiles des sous-familles F_x^3 (notons que dans ce cas le temps de résolution avec le modèle linéaire reste faible et proche de celui obtenu avec la méthode graphe). Le gain le plus intéressant est obtenu avec le modèle linéaire pour les familles les plus difficiles à résoudre c'est-à-dire les F_x^1.

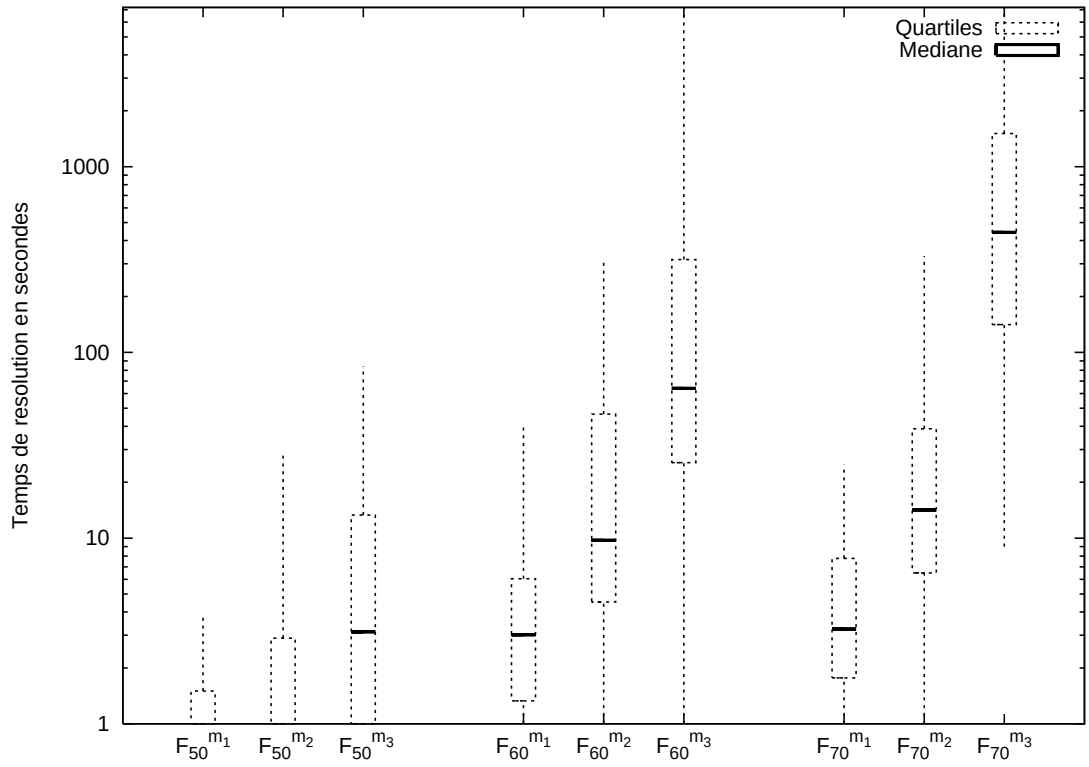


FIG. 6.18 – Influence de m_0 pour les familles de 50, 60 et 70 opérations

6.6 Comparaison avec une PSE

À présent, nous nous intéressons à la comparaison avec la PSE dédiée qui a été initialement proposée dans [DI05] pour la résolution du problème TLBP/B-M (voir section 3.6.2). Ce problème est une généralisation du TLBP/B-P, la procédure peut donc être appliquée dans le cadre du TLBP/B-P. Nous reprenons dans les figures 6.20, les résultats des expérimentations entreprises pour les instances décrites dans le tableau 6.3.

Dans les graphiques 6.20, les temps de calcul moyens obtenus avec la PSE sont notés *B&B* et ceux obtenus avec notre modèle linéaire orienté opérations sont marqués *CPLEX*. Rappelons que les familles de type F_x^1 ont une faible densité du graphe de précédence, il a été montré qu'elles sont les plus difficiles à résoudre (voir sous-section 6.4.2). Nous constatons pour ces familles que le modèle linéaire orienté opérations est souvent plus performant que la méthode dédiée, le gain le plus important est marqué pour certaines sous-familles de 60 et 70 opérations où Cplex permet de réduire le temps de calcul moyen par plus de la moitié. Concernant les sous-familles de type F_x^2 ayant une moyenne densité, le modèle linéaire présente des résultats équivalent à ceux de la PSE dédiée, néanmoins il peut être moins performant lorsque le nombre d'opérations devient trop important (70 et 80).

Afin de compléter l'analyse comparative des deux méthodes, nous présentons dans les figures 6.21 les temps minimum et maximum des instances. Nous constatons globalement

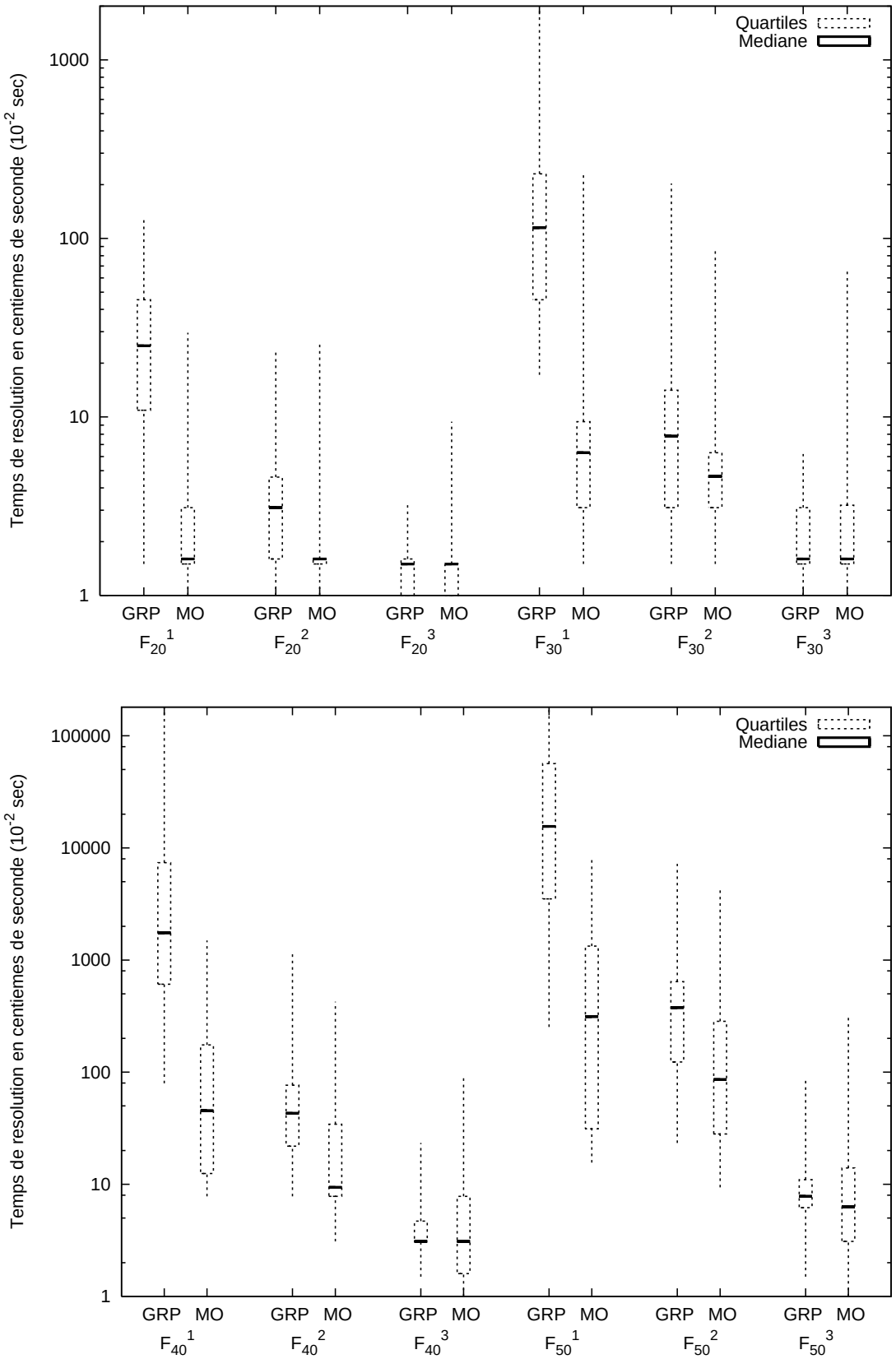


FIG. 6.19 – Temps de calcul de la méthode graphe et du modèle orienté opérations

le même comportement que pour les moyennes, c'est-à-dire que le modèle linéaire est plus performant sur les instances difficiles (F_x^1). Alors que la PSE dédiée est meilleure sur les instances moyennes F_x^2.

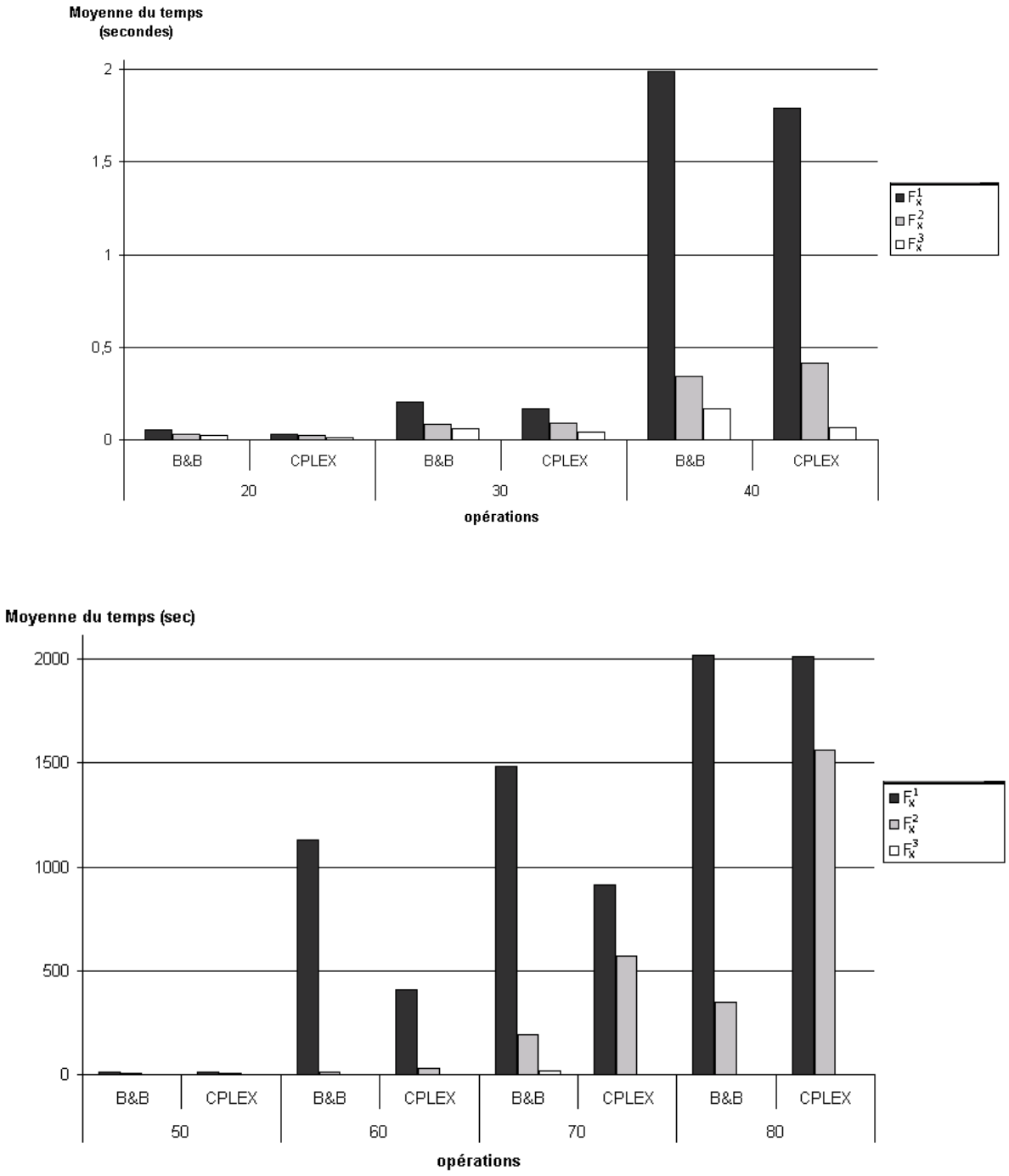


FIG. 6.20 – Temps moyens de la PSE et du modèle linéaire pour les familles de 20 à 80 opérations

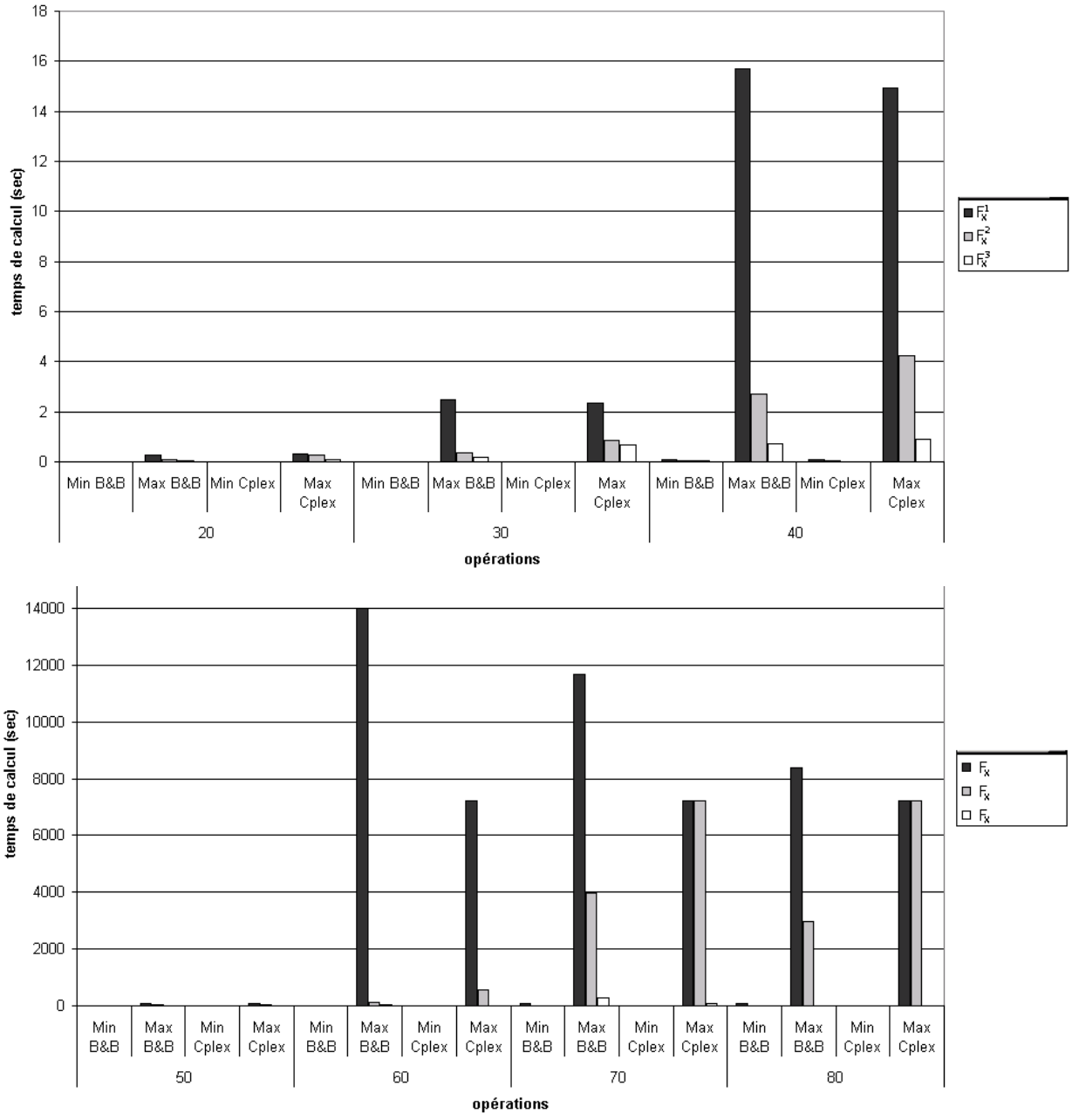


FIG. 6.21 – Temps de calcul de la PSE et du modèle orienté opérations

6.7 Conclusion

Dans ce chapitre, nous avons effectué de nombreux tests (près de 4000) pour évaluer les différents modèles proposés dans le chapitre précédent mais aussi pour mesurer l'apport des algorithmes de réduction du nombre de variables binaires. Les tests ont montré que les modèles de programmation linéaire en nombres entiers sont plus performants que les modèles de programmation par contraintes. Ils nous ont également permis de constater que les algorithmes de réduction du nombre de variables binaires permettaient de diminuer de façon significative le temps de calcul des instances.

Ensuite, nous avons comparé les performances des deux modèles linéaires en montrant que le modèle orienté opérations était plus efficace que le modèle orienté blocs. De plus, ces résultats confirment que l'utilisation d'outils tels que Cplex doit être accompagnée d'une bonne modélisation pour arriver à résoudre des instances difficiles. Ainsi, quand le modèle orienté blocs atteint ses limites pour des instances de 40 opérations, le modèle orienté opérations, permet de résoudre des problèmes ayant une taille de 80 opérations, doublant ainsi la taille des instances traitées. Ceci atteste de l'importance et du gain apporté par une modélisation exploitant la structure du problème. Cette phase d'expérimentation confirme ainsi qu'une coopération entre un modèle optimisé et un solver de qualité (utilisant ILOG Cplex) peut être aussi performante qu'une méthode dédiée telle que la méthode des graphes et la PSE.

Par ailleurs, nous avons également étudié l'influence de plusieurs paramètres tels que : le nombre de blocs, densité du graphe de précédence. L'étude a permis de déceler une corrélation entre le temps de calcul et les valeurs de ces paramètres.

Troisième partie

Quelques extensions du modèle TLBP/B-P

Chapitre 7

Mode d'activation mixte des unités d'usinage : TLBP/B-M

À présent, nous nous intéressons à la généralisation de notre approche au problème considérant un mode d'activation mixte dans le sens série-parallèle (voir formulation générique du problème dans [DIB05]). Dans ce type de lignes, une station correspond à plusieurs étages dont le fonctionnement est séquentiel de sorte que l'étage suivant ne peut être enclenché avant que le précédent n'ait terminé. Chaque étage peut contenir lui-même un sous-ensemble d'unités d'usinage qui sont activées en parallèle. Ainsi, lorsqu'il n'existe qu'un seul étage dans une station le problème se ramène au cas de l'activation parallèle, que nous avons étudié dans la seconde partie de ce mémoire. Par ailleurs, en présence d'une seule unité d'usinage par étage, le problème se ramène au cas séquentiel. Le cas mixte (TLBP/B-M) est donc une généralisation du cas parallèle (TLBP/B-P) et du cas séquentiel.

7.1 Description du TLBP/B-M

7.1.1 Les données

L'ensemble des opérations est également noté $\mathbf{N}$ pour désigner les opérations qui doivent être effectuées sur la ligne.

L'ensemble $\mathbf{B}$ est utilisé de la même manière que précédemment pour désigner les blocs disponibles. Ainsi, $\forall b \in \mathbf{B}, \mathcal{N}(b) \subseteq \mathbf{N}$ tel qu'il existe une unité d'usinage effectuant toutes les opérations de b. De même, chaque bloc b est caractérisé par son coût unitaire q_b et son temps d'exécution t_b.

Le coût moyen de l'ouverture d'une station est toujours noté C et chaque station peut contenir au maximum q_0 étages. Il est important de noter que le coût d'ouverture d'une station ne dépend pas du nombre d'étages qui seront effectivement utilisés.

La présence des étages activés de manière séquentielle nous amène à récrire les contraintes du problème en soulignant, à chaque fois, les différences avec le cas parallèle (TLBP/B-P).

7.1.2 Les contraintes

Le temps de cycle

La valeur T_0 est le temps de cycle maximum à ne pas dépasser. Le temps de cycle effectif de la ligne est déterminé par la station goulot[1]. Le temps de travail d'une station est égal au cumul des temps d'exécution des étages qui la composent en raison de leur exécution séquentielle. Par conséquent, la contrainte de temps de cycle T_0 doit être directement intégrée dans le modèle. Contrairement au cas parallèle, où il suffisait d'éliminer les unités d'usinage ayant un temps supérieur à T_0, pour le cas mixte l'activation séquentielle des étages rend ce pré-traitement nécessaire mais plus suffisant.

Les contraintes de précédence

Les contraintes de précédence entre les opérations sont définies de la même façon que pour le TLBP/B-P. Nous utilisons toujours D^{or} pour désigner les couples d'opérations qui sont reliées par une relation d'antériorité. À la différence du cas précédent, un couple d'opérations appartenant à D^{or} peut être affecté à la même station à condition que l'opération prédécesseur soit affectée à un étage antérieur à celui de l'opération successeur.

Les contraintes d'inclusion

Les contraintes d'inclusion pour les stations sont les mêmes, c'est-à-dire que certaines opérations doivent impérativement être effectuées sur la même station. Nous les représentons par la même collection D^{in}.

Les contraintes d'exclusion

De manière similaire au TLBP/B-P, nous désignons par D^{ex} la collection décrivant les contraintes d'exclusion entre les blocs qui ne peuvent pas être affectés à la même station.

Seulement, comme les unités d'usinage qui équipent le même étage sont activés en parallèle, il faut également considérer l'incompatibilité qui peut exister entre elles. Il faut par conséquent prendre en compte les interdictions d'affectation de certaines unités au même étage. Afin d'éviter toute confusion avec les contraintes d'incompatibilité aux stations, nous considérons les possibilités des blocs d'être mis en parallèle dans un même étage en les désignant par *contraintes de parallélisme*. Nous utilisons ainsi la collection D^{pb} dont les éléments sont des sous-ensembles de blocs, c'est-à-dire que si $p \in D^{pb}$, alors $p \subseteq \mathbf{B}$, tel que les blocs appartenant à p peuvent être placés sur le même étage.

[1]Rappelons que la station goulot est la station qui a le plus grand temps de travail parmi celles qui composent la ligne.

Limite sur le nombre de blocs

La limitation sur le nombre total d'unités d'usinage que peut comporter une station est donnée par le nombre n_0.

Limite sur le nombre d'étages

Comme nous l'avons déjà précisé, le nombre d'étages sur une même station est limité à q_0 au maximum. Ce nombre peut traduire une limitation physique due à la construction de la station où bien il est déterminé par le concepteur lui-même dont le savoir-faire lui permet de fixer cette valeur.

7.2 Un modèle linéaire en variables mixtes

Chaque station $k = 1, \ldots, m_0$ est composée de l'ensemble des étages potentiellement utilisables, soit $\{s_1^k, \ldots, s_{q_0}^k\}$. Du fait qu'un étage permet d'identifier la station à laquelle il appartient, il suffit de représenter la ligne par l'ensemble des étages de toutes les stations potentiellement ouvrables :

$$L = \{s_1^1, \ldots, s_{q_0}^1, \ldots, s_1^{m_0}, \ldots, s_{q_0}^{m_0}\}$$

De la même manière que pour le cas parallèle, une analyse préalable se basant sur les contraintes peut être faite afin de déduire les étages d'affectation dans lesquels, chaque bloc b, pourra être placé. Nous pouvons utiliser l'Algorithme 2 (voir section 5.4.2) pour obtenir l'étage au plus tôt et l'étage au plus tard afin de réduire le nombre d'affectations possibles.

Nous avons opté pour la généralisation de la formulation orientée opérations car elle a été la plus performante pour le cas parallèle. Ainsi, nous réutilisons $Q(i) = \{b \in \mathbf{B} | i \in \mathcal{N}(b)\}$ pour définir l'ensemble des blocs de $\mathbf{B}$ qui exécutent l'opération $i \in \mathbf{N}$.

7.2.1 Les variables de décision

Dans le cas présent, nous devons non seulement préciser l'affectation des blocs aux stations mais nous devons déterminer l'étage de la station pour chaque bloc affecté. De la même façon que pour le cas parallèle les variables positionnant l'ouverture des stations sont également employées.

L'affectation des blocs aux étages

Nous introduisons une variable x_{bjk} pour chaque bloc b afin de décider de son affectation à l'étage s_j^k, soit :

$$x_{bjk} = \begin{cases} 1 & \text{si le bloc } b \text{ est affecté à l'étage } s_j^k \\ 0 & \text{sinon} \end{cases}, \forall k = 1, \ldots, m_0, \forall j = 1, \ldots, q_0, \forall b \in \mathbf{B}$$

L'ouverture des stations

$$y_k = \begin{cases} 1 & \text{si la station } k \text{ est ouverte} \\ 0 & \text{sinon} \end{cases}, \forall k = m^* + 1, \ldots, m_0$$

par définition de la borne inférieure, pour tout $k \leq m^*$ (m^* est une borne inférieure sur le nombre de stations à ouvrir) les stations correspondantes sont assurément ouvertes. Il n'y a besoin donc que des variables de décision y_k pour $k = m^* + 1, \ldots, m_0$.

Les temps opératoires des étages

Afin d'exprimer les contraintes imposant le respect du temps de cycle maximum T_0, nous avons besoin d'introduire des variables additionnelles. Plus exactement, une variable réelle t_j^k est introduite, pour chaque étage s_j^k, $k = 1, \ldots, m_0, j = 1, \ldots, q_0$ exprimant son temps opératoire.

7.2.2 Formulation de l'objectif et des contraintes

La fonction objectif est donnée par (7.1), comme suit :

$$Minimiser \sum_{k=m^*+1}^{m_0} C y_k + \sum_{b \in \mathbf{B}} \sum_{k=1}^{m_0} \sum_{j=1}^{q_0} q_b x_{bjk} \tag{7.1}$$

Le premier terme de l'expression (7.1) fournit le coût engendré par la mise en place des stations supplémentaires à m^*. Du fait que les m^* premières stations participent systématiquement à la construction de toute solution réalisable et que leur coût est fixe, l'objectif (7.1) peut être alors exprimé uniquement en fonction du coût dû aux stations additionnelles qui seront ouvertes. Ainsi, pour retrouver le coût global de la ligne, il suffit de rajouter à la valeur de l'objectif le coût de l'ouverture des m^* premières stations. Quant au second terme de l'objectif, il correspond au coût des unités sélectionnées.

Les équations suivantes imposent l'unicité de l'exécution de chaque opération de $\mathbf{N}$:

$$\sum_{b \in Q(i)} \sum_{k=1}^{m_0} \sum_{j=1}^{q_0} x_{bjk} = 1, \quad \forall i \in \mathbf{N} \tag{7.2}$$

Pour assurer les contraintes de précédence, il suffit d'imposer l'exécution de toute opération prédécesseur dans un étage antérieur à celui de l'opération successeur :

$$\sum_{b\in Q(o)}\sum_{h=1}^{k-1}\sum_{l=1}^{q_0} x_{blh}+\sum_{b\in Q(o)}\sum_{l=1}^{j-1} x_{blk} \geq \sum_{r\in Q(o')} x_{rjk}, \quad \forall (o,o') \in D^{or}, \forall j=1,\ldots,q_0, \forall k=1,\ldots,m_0 \tag{7.3}$$

Les contraintes d'inclusion pour les stations doivent assurer l'exécution de certaines opérations au sein de la même station, elles sont exprimées comme suit :

$$\sum_{b\in Q(o)}\sum_{l=1}^{q_0} x_{slk} = \sum_{r\in Q(o')}\sum_{j=1}^{q_0} x_{rjk}, \quad \forall (o,o') \in D^{in}, \forall k=1,\ldots,m_0 \tag{7.4}$$

Les contraintes d'exclusion pour les blocs interdisant leur affectation dans une même station :

$$\sum_{b\in E}\sum_{j=1}^{q_0} x_{bjk} \leq |E|-1, \forall E \in D^{ex}, \forall k=1,\ldots,m_0 \tag{7.5}$$

La limitation sur le nombre de blocs par chaque station est imposée en utilisant les inégalités (7.6) :

$$\sum_{b\in \mathbf{B}}\sum_{j=1}^{q_0} x_{bjk} \leq n_0, \quad \forall k=1,\ldots,m_0 \tag{7.6}$$

Les sous-ensembles $p \in \overline{D^{pb}}$ correspondent aux blocs qui ne peuvent pas être mis en parallèle sur un même étage.

$$\sum_{b\in p, b\neq s} x_{bjk} \leq 1-x_{sjk}, \quad \forall s \in p, \forall p \in \overline{D^{bp}}, \forall k=1,\ldots,m_0, \forall j=1,\ldots,q_0 \tag{7.7}$$

Les inéquations (7.8) expriment le temps opératoire d'un étage. Plus exactement, le temps d'exécution de chaque étage est déterminé par le plus grand des temps d'exécution parmi les temps des blocs qui lui sont affectés :

$$t_b x_{bjk} \leq t_j^k, \quad \forall b \in \mathbf{B}, \forall k=1,\ldots,m_0, \forall j=1,\ldots,q_0 \tag{7.8}$$

Afin que le temps de travail de chaque station respecte la borne T_0 sur le temps de cycle, nous imposons, pour chaque station, que le cumul des temps opératoires des étages qui la composent soit inférieur à la limite T_0 :

$$\sum_{j=1}^{q_0} t_j^k \leq T_0, \quad \forall k=1,\ldots,m_0 \tag{7.9}$$

Comme pour le modèle précédent, nous introduisons les inéquations (7.10) pour écarter les solutions où des stations intermédiaires sont vides.

$$y_{k-1} - y_k \geq 0, \quad k = m^* + 2, \ldots, m_0 \tag{7.10}$$

Les contraintes (7.11) permettent d'inférer les décisions prises concernant l'ouverture des stations dans la fonction objectif. Autrement dit, dés qu'un bloc est affecté à la station courante la variable de décision y_k correspondante est positionnée à 1, ce qui permet de répercuter le coût de la nouvelle station créée dans l'objectif.

$$y_k \geq x_{bjk}, \quad \forall b \in \mathbf{B}, \forall k = m^* + 1, \quad \forall j = 1, \ldots, q_0 \tag{7.11}$$

7.3 Perspectives

Nous avons implémenté ce modèle et entamé une phase expérimentale qui a permis de relever des premiers résultats [BDDI06]. Près de 200 instances ont été testées pour évaluer les performances de la formulation, toutefois nous avons fait le choix de ne pas rapporter ces résultats car il sont à compléter. Il a été notamment intéressant de constater que les instances du TLBP/B-M requièrent un temps de calcul plus important par rapport à celles de TLBP/B-P. En effet, la présence des étages multiplie le nombre d'affectations pour chacun des blocs par rapport au TLBP/B-P où il n'y qu'une affectation possible par station. De plus, l'introduction des nouvelles contraintes de parallélisme rend le problème plus difficile ralentissant globalement la résolution des instances. C'est pour améliorer ces performances que nous suggérons d'intégrer des coupes pour réduire le domaine des solutions réalisables. En particulier, nous pensons réduire davantage le nombre de solutions symétriques en introduisant des coupes pour favoriser l'affectation des blocs aux premiers étages. Ainsi, il ne faut autoriser l'ouverture d'un étage ultérieure que lorsque l'ensemble des étages qui le précèdent ont été utilisés. De plus, des coupes développées pour le problème de set partitioning peuvent être intéressantes à tester. De la même façon que pour le cas du TLBP/B-P, une étude de l'influence des paramètres considérant la densité du graphe de parallélisme et le nombre d'étages par stations est à envisager.

Chapitre 8

Généralisation au problème multi-produit

Dans le précédent chapitre, nous avons apporté une première extension du modèle linéaire orienté opérations pour le TLBP/B-P. À présent, nous montrons une seconde généralisation, qui se situe, cette fois-ci, du point de vue du nombre de produits usinés. Plus exactement, nous montrons comment un modèle pour la configuration des lignes mono-produit peut être étendu à un modèle pour configurer une ligne multi-produit. Il s'agit de configurer la ligne de manière à ce que plusieurs types de produits, nécessitant chacun un ensemble propre d'opérations, puissent être usinés. Toutefois, il est nécessaire que les produits à usiner soient des produits proches du point de vue des caractéristiques d'usinage. En particulier, il est supposé qu'ils ont un certain nombre d'opérations communes pour leur fabrication [KL98]. Nous précisons, dans la suite, les conditions dans lesquelles une telle adaptation est envisageable.

Chaque produit est caractérisé par un ensemble des données comportant notamment l'ensemble des opérations à effectuer. De plus, les contraintes relatives aux opérations sont également propres à chaque type de produits, ainsi, les précédences et les contraintes d'inclusion qui sont données pour chaque type. Les contraintes relatives aux unités d'usinage, quant à elles, sont communes à la famille des produits. La démarche proposée revient à agréger l'ensemble des produits pour les considérer comme un produit unique. De cette manière, nous pourrons appliquer le modèle mono-produit au produit agrégé. Dans la section qui suit, nous décrivons plus précisément les hypothèses que nous avons adoptées dans ce cas.

8.1 Hypothèses

Les hypothèses inchangées par rapport à celles prises pour le TLBP/B-P, sont les suivantes :

1. Une opération ne doit pas être exécutée plus d'une seule fois par les unités sélectionnées sur la ligne,

2. Les coûts des stations et des unités d'usinage, ainsi que les temps d'exécution sont connus.
3. les unités d'usinages disponibles sont connues à l'avance, toutefois à la différence des problèmes précédents, le choix d'une unité n'implique pas forcément l'exécution de toutes les opérations qu'elle est capable d'exécuter. Ainsi, nous autorisons l'exécution partielle des opérations. En pratique, cette hypothèse est applicable dans le cas des machines reconfigurables dont les broches peuvent être modifiées (ajoutés ou supprimés).
4. les contraintes de précédence sont *non strictes* (de type $\leq$), cette hypothèse résulte de la prise en compte des circuits dans le graphe de précédence agrégé. Plus exactement, nous transformons les opérations qui constituent un circuit en macro-opération qui sera alors effectuée par un outil combiné (outil à étages). C'est pourquoi les contraintes de précédence qui relient les opérations initiales doivent être moins strictes car nous autorisons leur exécution en même temps. Dans le cas non strict, l'interdiction imposée par une contrainte de précédence du couple (i, j) porte sur l'impossibilité d'exécuter j avant i uniquement, ce qui fait qu'il est possible d'exécuter i avant j ou bien en même temps. Il est important de signaler que nous pouvons également considéré le cas stricte en éliminant de $\mathbf{B}$ tous les blocs b contenant simultanément les deux opérations i et j.

Dans la section qui suit, nous introduisons quelques notations nécessaires à la formulation du modèle multi-produits.

8.2 Notations

1. P est une famille de produits à fabriquer dont le nombre de produits est donnée par sa cardinalité $|P|$,
2. N_p est l'ensemble des opérations du produit $p = 1, \ldots, |P|$,
3. $\mathbf{N} = \bigcup_{p=1}^{|P|} \mathrm{N}_p$ est l'ensemble des opérations de la famille qui comporte tous les types de produits,
4. $\mathbf{B}$ est l'ensemble des blocs disponibles pour effectuer l'ensemble des opérations $\mathbf{N}$. En pratique, il est obtenu en fusionnant l'ensemble des blocs pour chaque type de produit et peut être éventuellement complété en ajoutant des outils combinés,
5. q_b est le coût du bloc b et t_b est son temps d'exécution,
6. C est coût moyen de l'ouverture d'une station,
7. m_0 est le nombre maximal de stations possibles et n_0 est le nombre maximal de blocs par station,
8. $\triangle_t$ est l'interval de temps pour lequel la demande en produits de type p est connue et égale à d_p, $p = 1, \ldots, |P|$.

Afin de considérer les contraintes de tous les types de produits nous proposons de les agréger. Dans la section qui suit nous expliquons en détails la démarche adoptée.

8.3 L'agrégation des contraintes

L'agrégation des contraintes est un des moyens possibles pour la prise en compte simultanée de l'ensemble des contraintes qui concernent tous les types de produits. Nous l'avons appliquée pour obtenir l'ensemble total des contraintes de précédence et d'inclusion qui concernent les opérations. Suite à cette étape, nous déduisons des blocs à supprimer, au vue d'hypothèses que nous avons fournies, et finalement nous mettons à jour l'ensemble total des contraintes d'exclusion pour rester cohérent avec la réduction qui a été faite. Dans ce qui suit, nous décrivons d'abord de façon plus précise l'agrégation des contraintes liant les opérations et ensuite nous indiquons les substitutions nécessaires pour l'ensemble de blocs et des contraintes d'exclusion.

8.3.1 Les contraintes de précédence

Pour chaque type de produit $p = 1, \ldots, |P|$, il existe un graphe de précédence $G_p^{or} = (\mathbf{N}_p, D_p^{or})$. Afin que l'ensemble des contraintes de la famille P soient respectées il suffit de fusionner l'ensemble des graphes de précédence pour obtenir un graphe total (agrégé) :

$$G^{or} = \bigcup_{p=1}^{|P|} G_p^{or},$$

Comme nous l'avons déjà évoqué, les contraintes de précédence sont supposées être *non strictes*, dans ce cas l'exécution d'opérations liées par une contrainte de précédence peut se faire en parallèle. Plus exactement, s'il existe $(i, j) \in D_p^{or}$ alors il y a deux possibilités d'exécution pour la paire (i, j), à savoir : l'opération i est exécutée strictement avant l'opération j ou celles-ci sont exécutées simultanément par une même tête d'usinage. Cela se traduit par le fait d'autoriser l'utilisation d'unités d'usinage avec outils étagés pouvant effectuer plusieurs opérations liées par des contraintes de précédence.

Le graphe global engendré par la fusion des graphes de précédence des différents types doit être cohérent en ce sens qu'il ne doit pas comporter de circuit.

Le graphe global G^{or} est construit, plus exactement, en appliquant les règles suivantes :

1. tout sommet appartenant à un des graphes G_p^{or} est rapporté dans le graphe global G^{or} ainsi que l'ensemble de ses arcs,

2. les redondances sont éliminées en supprimant tout arc reliant deux sommets pour lesquels il existe déjà un chemin,

3. les circuits doivent être identifiés et les opérations qui les composent sont considérées comme des macro-opérations qui sont remplacées par de simples sommets (il faut vérifier qu'il existe des blocs exécutant ces macro-opérations).

8.3.2 Les contraintes d'inclusion et d'exclusion

Pour les contraintes d'inclusion chaque type p est caractérisé par un ensemble $D_p^{in}, p = 1, \ldots, |P|$. L'agrégation est effectuée en fusionnant tous les ensembles D_p^{in} pour obtenir un ensemble global D^{in}. La fusion des opérations formant un circuit dans le graphe global G^{or} imposent des modifications sur l'ensemble des autres contraintes, notamment, sur l'ensemble des contraintes d'inclusion. Plus précisément, lorsqu'une opération est à l'origine d'un circuit et qu'elle appartient désormais à la macro-opération, de plus si celle-ci est contenue dans un sous-ensemble d'inclusion quelconque alors ce sous-ensemble d'inclusion est complété avec les opérations manquantes de la macro-opération. De la même manière que dans le modèle mono-produit, il est alors possible de réduire la taille des éléments de l'ensemble D^{in} en fusionnant en un seul sous-ensemble tous ceux qui ont une intersection non vide.

Comme pour les contraintes d'inclusion, l'ensemble total D^{ex} est obtenu en fusionnant l'ensemble des exclusions de chaque type, soit : $D^{ex} = \bigcup_{p=1}^{|P|} D_p^{ex}$

Il est également possible de réduire l'ensemble agrégé d'exclusion en éliminant les contraintes qui concernent les blocs qui ont été supprimés.

8.3.3 L'ensemble des blocs

Cet ensemble doit contenir l'ensemble des unités pour tous les types de produits, c'est donc l'union de tous les ensembles de blocs disponibles, $\mathbf{B} = \bigcup_{i=1}^{|P|} \mathbf{B}_i$

De cet ensemble il faut supprimer tous les blocs qui n'exécutent qu'un sous-ensemble des opérations d'une macro-opération détectée précédemment. Des blocs complémentaires peuvent éventuellement être ajoutés pour les macro-opérations qui ne sont contenues dans aucun bloc existant (si c'est possible), autrement le problème serait irréalisable.

8.4 Un exemple

Nous reprenons l'exemple présenté dans notre article [BBD06] pour illustrer l'ensemble des notions introduites pour le modèle multi-produit. Pour commencer, nous décrivons les données et contraintes du problème puis nous déroulons la phase d'agrégation en appliquant les différents pré-traitements proposés.

8.4.1 Description des données et contraintes

Pour cet exemple, la famille comporte trois types de produits ($|P| = 3$), notés : p_1, p_2 et p_3. L'ensemble des opérations à effectuer, pour chaque type, est donné dans le tableau 8.1.

p_i	$\lvert N_i \rvert$	N_i
p_1	10	$\{1,2,3,4,5,6,7,8,9,10\}$
p_2	12	$\{1,2,3,4,5,6,7,8,9,10,11,12\}$
p_3	10	$\{1,2,5,7,8,9,10,11,12,13\}$

TAB. 8.1 – L'ensemble des opérations pour la famille de produits

L'ensemble total des opérations est donné par $\mathbf{N} = \bigcup_{i=1}^{|P|} N_i = \{1,\ldots,13\}$, et les opérations communes sont $\bigcap_{i=1}^{|P|} N_i = \{1,2,5,7,8,9,10\}$.

Pour chaque type p_i, un graphe de précédence G_i^{or} est fourni, nous les rapportons dans la figure 8.1.

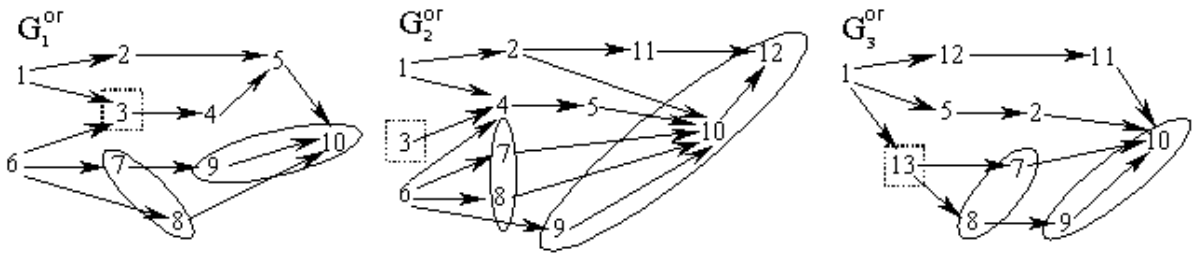


FIG. 8.1 – Les graphes de précédence de la famille de produits.

Pour chaque type de produit p_i, l'ensemble des contraintes d'inclusion est donné par D_i^{in}, comme suit :

$D_1^{in} = \Big\{\{7,8\},\{9,10\}\Big\}$, $D_2^{in} = \Big\{\{7,8\},\{9,10,12\}\Big\}$ et $D_3^{in} = \Big\{\{7,8\},\{9,10\}\Big\}$

De plus, les opérations $D_s^{in} = \Big\{\{3,13\}\Big\}$ doivent être exécutées sur la même station. Ces contraintes doivent être considérées en plus des contraintes d'inclusion de chaque type car elles n'ont pas pu être considérées dans un des D_i^{in}. Plus exactement, aucun des ensemble N_i ne contient les deux opérations en même temps. Ainsi, pour le type p_1 et p_2 c'est seulement l'opération 3 qui est présente dans N_1 tandis que pour p_3 c'est uniquement l'opération 13 qui est considérée.

Concernant les contraintes d'exclusion elles sont données par un seul ensemble D^{ex} car elles concernent les blocs disponibles et sont, de ce fait, communes à tous les types de produits. Elles sont décrites comme suit :

$D^{ex} = \Big\{\{b_1,b_6,b_7\},\{b_1,b_9\},\{b_5,b_{11}\}\Big\}$

Les blocs disponibles constituant $\mathbf{B}$ sont décrits dans le tableau 8.2.

b	opérations	temps opératoire (t_b)	coût (q_b)
b_1	{1,3,6,13}	9	250
b_2	{1,3,13}	8	170
b_3	{ 1,2,7,8}	6	281
b_4	{2,5,9}	10	150
b_5	{2,5,7,8,11}	9	275
b_6	{ 2,6,9,10}	11	230
b_7	{4,6,8,10}	13	211
b_8	{4,7,8}	9	160
b_9	{4,7,8,9}	10	215
b_{10}	{5,12,13}	6	158
b_{11}	{10,11,12,13}	12	230
b_{12}	{2,5,10,11,12}	11	260

TAB. 8.2 – L'ensemble des blocs pour la famille de produits

Les temps de cycle T_{0_i}, pour chaque type p_i, sont déduits de la période commune $\triangle_t = 48300$ et de la demande pour chaque type, donnée respectivement par : $d_1 = 2100$, $d_2 = 1932$ et $d_3 = 2300$, en utilisant l'expression $T_{0_i} = \triangle_t / d_i$. Ainsi, $T_{0_1} = 23$, $T_{0_2} = 25$ et $T_{0_3} = 21$.

8.4.2 Agrégation et pré-traitements

L'agrégation des contraintes de précédence de l'ensemble des graphes présentées dans la figure 8.1 permet de détecter deux circuits, à savoir : $2 \mapsto 5 \circlearrowleft$ et $11 \mapsto 10 \mapsto 12 \circlearrowleft$. Chaque circuit est supprimé et remplacé par une macro-opérations substituant aux opérations formant le circuit. Dans ce cas, nous obtenons deux macro-opérations que nous désignons par les sommets $a = \{2, 5\}$ et $b = \{10, 11, 12\}$ dans le graphe agrégé de la figure 8.2.

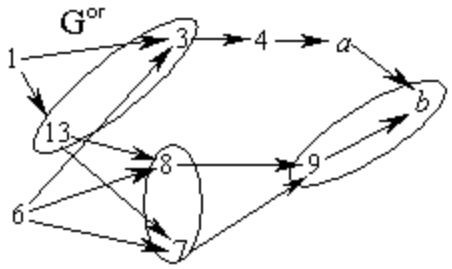


FIG. 8.2 – Le graphe de précédence pour la famille de produits.

L'introduction des macro-opérations engendrent des modifications quant à la composition de l'ensemble des opérations de chaque type de produit. Nous indiquons ces repercussions dans le tableau 8.3

La présence de ces macro-opérations engendre également des mises à jour de l'ensemble des blocs disponibles **B**. Plus précisément, nous appliquons les deux règles suivantes :

p_i	$\lvert\mathbf{N}_i\rvert$	$\mathbf{N}_i$
p_1	9	$\{1, a, 3, 4, 6, 7, 8, 9, b\}$
p_2	9	$\{1, a, 3, 4, 6, 7, 8, 9, b\}$
p_3	7	$\{1, a, 7, 8, 9, b, 13\}$

TAB. 8.3 – La modification de l'ensemble des opérations pour la famille de produits

1. chaque bloc ne considérant qu'un sous-ensemble d'une macro-opération quelconque est à supprimer de l'ensemble $\mathbf{B}$.
2. lorsqu'il existe un circuit dans le graphe de précédence reliant les blocs, il faut éliminer ce circuit en supprimant un des blocs qui le forme. Nous avons fait le choix de supprimer le bloc le plus coûteux par opération, toutefois nous pourrions également appliquer d'autres règles si le problème devient infaisable, par exemple, ne pas supprimer les blocs qui sont l'unique possibilité d'exécution de certaines opérations.

L'application de la première règle sur l'ensemble $\mathbf{B}$ induit l'élimination des blocs b_3, b_5, b_6, b_7 et b_{10}.

Quant à l'application de la seconde règle, elle implique d'éliminer un des blocs qui forment le circuit montré dans la figure 8.3.

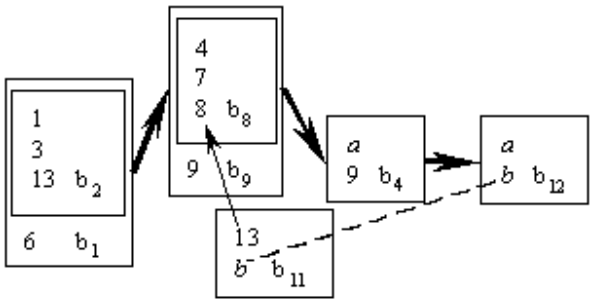


FIG. 8.3 – Le graphe de précédence pour les blocs.

Le bloc choisi parmi b_4, b_8, b_{11} et b_{12} est b_{11} car c'est celui dont le coût par opération est le plus élevé (57,5). Ce qui ramène l'ensemble des blocs disponibles à :

$\mathbf{B} = \{b_1, b_2, b_4, b_8, b_9, b_{12}\}$.

L'ensemble des contraintes d'inclusion doit également être mis à jour en considérant les macro-opérations.

$D_1^{in} = \Big\{\{7,8\},\{9,b\}\Big\}$, $D_2^{in} = \Big\{\{7,8\},\{9,b\}\Big\}$ et $D_3^{in} = \Big\{\{7,8\},\{9,b\}\Big\}$

L'union de l'ensemble de ces contraintes plus la contrainte additionnelle entre produits :

$D^{in} = \Big\{\{7,8\},\{9,b\},\{3,13\}\Big\}$.

De la même manière que pour les autres ensembles, les contraintes d'exclusion doivent

être modifiées dans le sens où il faut prendre en compte l'élimination de certains blocs. Du fait qu'un sous-ensemble d'exclusion traduit l'interdiction de l'affectation simultanée des blocs correspondant, il faut alors supprimer tous les sous-ensembles de blocs faisant intervenir un des blocs supprimés.

Pour cet exemple, cela revient à garder uniquement $\{b_1, b_9\}$ dans D^{ex}.

8.5 Conclusion

L'intérêt du problème multi-produit que nous avons étudié dans le présent chapitre est incontestable. Malgré cette réalité industrielle, ce problème n'a pas encore été investi dans le contexte de l'usinage ; notre contribution s'inscrit dans ce contexte en généralisant les travaux effectués dans la seconde partie de cette thèse. En effet, le noyau de l'approche proposée pour le TLBP/B-M ou encore pour le problème multi-produit réside dans la formulation du modèle pour le TLBP/B-P étudié dans la seconde partie de ce mémoire. Plusieurs extensions deviennent alors possibles même si cela demande parfois de redéfinir certaines hypothèses. En particulier, nous avons supposé que les contraintes de précédence dans le cas multi-produit ne sont plus au sens strict telles qu'elles ont été définies pour le TLBP/B-P, et ce en raison de la présence de macro-opérations introduites suite à l'agrégation des contraintes. En effet, cette considération au sens non strict permet de contourner le problème de circuit qui peut se poser en envisageant d'utiliser des outils combinés capables d'effectuer certaines opérations liées par des précédences sans les dissocier.

Conclusion générale et perspectives

Les travaux effectués dans cette thèse s'inscrivent dans le cadre de l'étude et du développement des systèmes d'aide à la décision pour la configuration des lignes d'usinage à partir d'équipements standard. Plus précisément, la démarche proposée fournit au concepteur des outils efficaces, fondés sur des méthodes d'optimisation pour répondre au problème de conception et reconfiguration de ces lignes.

Nous avons souligné la complexité du problème posé et les caractéristiques essentielles de ces lignes imposant la recherche de nouvelles approches pour le problème de leur configuration. Nous avons évoqué notamment les regroupements des opérations en blocs généralisant ainsi le problème d'équilibrage de lignes d'assemblage (SALBP). Ces blocs correspondent physiquement à des unités d'usinage standard effectuant chacune un sous-ensemble d'opérations. L'ensemble de ces unités est déterminé suite à une étude réalisée en amont. La nécessité de la prise en compte simultanée de tous ces éléments a engendré une difficulté supplémentaire par rapport aux travaux déjà étudiés dans la littérature. Ainsi, les méthodes existantes ne sont pas à même de résoudre ce nouveau problème.

Cette thèse a permis de proposer une modélisation efficace pour ce nouveau problème, mais aussi plusieurs algorithmes améliorant les performances des approches suggérées.

Dans la seconde partie de ce mémoire, nous avons testé une approche basée sur la programmation par contraintes. Nous avions opté pour l'approche PPC en raison de la nature fortement contrainte du problème traité et de la disponibilité d'outils tels que ILOG Solver. Dans ce cadre, nous avons apporté une formulation générique et des règles de propagation de certaines contraintes. Suite aux tests expérimentaux effectués, l'approche PPC s'est avérée insuffisante pour la résolution des instances de moyenne et grande taille. En effet, l'approche a atteint ses limites assez rapidement même si elle apporte des solutions réalisables assez vite. Nous pensons que ce comportement est dû à la nature de l'objectif qui est une somme pondérée de plusieurs variables. Nous avons proposé d'utiliser une borne de type relaxation linéaire en appliquant la propagation des différentes contraintes du problème. Au vu des résultats, ce schéma doit être amélioré en envisageant de définir une contrainte globale pour permettre une meilleure propagation sur l'objectif. En outre, nous pensons qu'il faudrait développer nos propres outils pour appliquer la formulation et les techniques de propagation suggérées. En effet, ILOG Solver ne permet pas de gérer l'espace mémoire lors de la construction de l'arbre de branchements, ce qui réduit la taille des instances qui peuvent être traitées.

Ensuite, nous nous sommes intéressés à la programmation linéaire en nombres entiers pour laquelle nous avons proposé deux modèles. Le premier modèle est orienté blocs car

les contraintes sont exprimées en fonction des blocs d'opérations. Quant au modèle orienté opérations, il est formulé en tenant compte de la nature des contraintes, c'est-à-dire que les contraintes qui sont relatives aux opérations sont exprimées en fonction de ces dernières. Dans les deux cas, nous avons proposé d'abord d'analyser les données afin d'optimiser leur représentation. Pour ce faire, nous avons introduit différents éléments de modélisation avant d'affiner la formulation du modèle final.

Par la suite, nous avons proposé différents algorithmes pour la réduction de la taille des modèles. Plus précisément, ces algorithmes exploitent la structure des contraintes du problème pour en déduire des impossibilités d'affectation pour chaque bloc d'opérations. Ceci revient à définir un intervalle contenant les indices des stations dans lesquelles chaque bloc peut être assigné. Ainsi, en réduisant ces intervalles nous avons diminué le nombre de variables binaires des modèles. Nous avons montré dans la phase expérimentale l'apport de ces algorithmes qui ont permis de réduire de façon considérable les temps de calcul. Il est également intéressant de signaler que les algorithmes les plus spécifiques sont les plus performants en termes de réduction du nombre de variables, mais pas forcément en termes de temps de calcul. Ainsi, nous avons opté pour l'Algorithme 2 car il a un bon rapport nombre de variables éliminées / temps de calcul.

Dans la phase expérimentale, nous avons réalisé d'abord une analyse comparative entre les deux modèles linéaires pour en évaluer les performances. Ces résultats ont permis de constater la supériorité du modèle orienté opérations résolvant le plus grand nombre d'instances dont la taille atteignait les 80 opérations. Le modèle a permis notamment de réduire de façon significative le temps de résolution pouvant le diviser par 100, voire plus pour certaines instances de grande taille. Ainsi, l'intérêt de la formulation orientée opérations est d'autant plus important pour les familles difficiles.

Nous avons poursuivi l'étude du modèle orienté opérations pour mesurer l'impact de certaines caractéristiques du problème sur les performances du modèle. Il a été décelé notamment que les instances les plus difficiles à résoudre étaient celles ayant une faible densité du graphe de précédence. Nous pensons que cela est dû au fait que le nombre de permutations pour l'affectation des blocs devient important lorsque la densité est faible. Nous avons également montré que le nombre de blocs disponibles a une influence forte vu que les instances ayant le nombre le plus important de blocs ont requis jusqu'à 100 fois plus de temps de calcul. Pour finir, nous avons étudié l'impact du nombre d'opérations par bloc et du nombre de stations maximum. Cette étude a également permis de conclure que les valeurs numériques de ces caractéristiques avaient une incidence directe sur le temps de calcul.

Dans la troisième partie de cette thèse, nous avons présenté deux extensions du problème initial. Dans la première extension, nous avons suggéré une généralisation du modèle orienté opérations pour le problème de configuration des lignes dont le mode d'activation des unités est mixte. Ce problème est plus complexe que le précédent car plusieurs unités d'usinage peuvent être mise en parallèle dans un même étage et une station peut être composées de plusieurs étages séquentiels. L'étude de ce mode a nécessité l'introduction de nouvelles contraintes pour tenir compte des incompatibilités d'unités pouvant être affectées à un même étage. Ensuite, une seconde extension a été étudiée en adaptant le modèle orienté opérations pour les lignes multi-produit. L'approche qui a été proposée se base sur l'agrégation des

contraintes et données pour chaque type de produit.

Nous avons ainsi montré que l'étude du problème initial (noté TLBP/B-P) considérant le mode d'activation parallèle a rendu possible différentes extensions définissant des problèmes plus complexes tels que celui où le mode d'activation est mixte pour les unités d'usinage ou encore celui considérant plusieurs types de produits dans la ligne. Les approches proposées pour la modélisation et la résolution de ces extensions s'appuient ainsi essentiellement sur le modèle proposé pour le problème initial.

Par ailleurs, ces travaux ont permis d'ouvrir de nouvelles perspectives pour les études futures. Par exemple, il serait notamment intéressant d'introduire le TLBP/B2, par analogie à la notation du SALBP-2, où l'objectif à minimiser serait le temps de cycle au lieu du coût de la ligne. Cette étude pourrait s'inscrire, comme pour le SALBP, dans une approche incrémentale globale optimisant d'abord le temps de cycle en considérant le TLBP/B2, ce qui permet d'obtenir une valeur du temps de cycle pour pouvoir lancer, ensuite, l'étude du TLBP/B optimisant le coût. De plus, une autre généralisation de ces deux problèmes peut être envisagée où le critère à optimiser serait l'efficacité de la ligne (définie dans ce cas par la minimisation du temps de cycle et du coût).

L'intérêt pratique du TLBP/B2 est incontestable particulièrement pour les lignes à reconfigurer. Plus exactement, lorsque la ligne est déjà mise en service et que la demande croît, il faut alors reconfigurer cette ligne de manière à minimiser le temps de cycle. Ainsi, à partir des unités et stations mises en place, il s'agit bien de retrouver une structure de ligne minimisant le temps de cycle.

Pour conclure, nous avons proposé des outils d'aide à la décision basés sur des méthodes de programmation mathématique pour la configuration des lignes d'usinage. Qu'il s'agisse de conception préliminaire ou de reconfigurations de ces lignes, l'intérêt industriel et les enjeux économiques sont capitaux. Nous avons souligné notamment les sommes colossales investies dans la mise en place de ce type de lignes, qui se chiffrent en millions d'euros, rendant l'usage d'outils d'optimisation indispensable. D'autre part, le gain réalisé grâce à l'emploi d'une telle démarche se répercute directement sur le coût de revient unitaire, aussi une étude minutieuse devient stratégique à ce stade.

Notations

Pour une ligne de transfert

T	temps de cycle effectif de la ligne
T_0	temps de cycle maximum de la ligne
m	nombre de stations composant la ligne
m_0	nombre maximum de stations dans la ligne
m^*	borne inférieure sur le nombre de stations du $TLBP$

Pour les stations

S_k	la $k^{\text{ème}}$ station de la ligne
C	coût moyen de la mise en place d'une station
n_0	nombre maximum d'unités d'usinage par station
y	variable de décision déterminant le nombre de stations établies (modèle orienté blocs)
y_k	variable binaire déterminant l'ouverture ou non de la $k^{\text{ème}}$ station
TS_k	le temps de travail de la $k^{\text{ème}}$ station pendant chaque cycle

Pour les étages

q_0	le nombre maximum d'étages par station
s_j^k	le $j^{\text{ème}}$ étage de la $k^{\text{ème}}$ station
Ts_j^k	le temps de travail du $j^{\text{ème}}$ étage la $k^{\text{ème}}$ station pendant chaque cycle
t_j^k	la variable de décision délimitant le temps de travail effectif du $j^{\text{ème}}$ étage de la $k^{\text{ème}}$ station

Pour les opérations

i, j, o, o'	des opérations
$\mathbf{N}$	ensemble des opérations
D^{or}	ensemble des contraintes de précédence entre les opérations
$\Gamma^-(b)$	ensemble des opérations prédécesseurs du bloc b
D^{in}	collection de sous-ensembles représentant les contraintes d'inclusion
d	sous-ensemble de contraintes d'inclusion appartenant à D^{in}
G^{or}	graphe de précédence pour les opérations
G^{or}_{inv}	graphe de précédence inversé pour les opérations
g	un niveau d'opérations
$rank(j)$	rang de l'opération j dans le graphe de précédence G^{or}
$\overline{rank}(j)$	rang de l'opération j dans le graphe de précédence inversé G^{or}_{inv}

G^{in}	graphe d'inclusion pour les opérations

Pour les blocs

$\mathbf{B}$	ensemble des blocs disponibles (unités d'usinage)
B, B'	sous-ensembles de blocs disponibles appartenant à $\mathbf{B}$
b, b'	des blocs appartenant à $\mathbf{B}$
t_b	temps opératoire du bloc b
q_b	coût du bloc b
$\mathcal{N}(b)$	ensemble des opérations contenues dans le bloc b
x_{bk}	variable de décision pour l'affectation du bloc b à la station k
x_{bjk}	variable de décision pour l'affectation du bloc b au $j^{\text{ème}}$ étage de la $k^{\text{ème}}$ station
D^{ex}	ensemble des contraintes d'exclusion (incompatibilité)
E	sous-ensemble de blocs incompatibles
D^{pb}	collection des contraintes de parallélisme
G^{EB}	graphe de précédence pour les blocs
$head_b$	la station au plus tôt du bloc b
$tail_b$	la station au plus tard du bloc b

Bibliographie

[Aig95] M. Aigner. *Turán's Graph Theorem*. Amer. Math. Monthly 102, 1995.

[Ame00a] M. Amen. An exact method for cost-oriented assembly line balancing. *International Journal of Production Economics*, 64 :187–195, 2000.

[Ame00b] M. Amen. Heuristic methods for cost-oriented assembly line balancing: a survey. *International Journal of Production Economics*, 68 :1–14, 2000.

[Ame01] M. Amen. Heuristic methods for cost-oriented assembly line balancing: a comparaison on solution quality and computing time. *International Journal of Production Economics*, 69 :255–264, 2001.

[Arc66] A.L. Arcus. COMSOAL: a computer method of sequencing operations for assembly lines. *International Journal of Production Research*, 4 :259–277, 1966.

[AS93] R.G. Askin and C.R. Standridge. *Modeling and Analysis of Manufacturing Systems*. John Wiley & Sons, Inc, 1993.

[Bay86] I. Baybars. A survey of exact algorithms for the simple line balancing problem. *Management Science*, 32 :909–932, 1986.

[BBD06] S. Belmokhtar, A.I. Bratcu, and A. Dolgui. *Modular Machining Line Design and Reconfiguration*: *Some Optimization Methods. In Manufacturing the Future*: *Concepts-Technologies-Visions*, pages 125 –152. 3 -86611-198-3. pro literatur Verlag, Mammendorf, Germany, 2006.

[BD62] R. Bellman and S.E. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, 1962.

[BDDI06] S. Belmokhtar, A. Dolgui, X. Delorme, and I Ihnatsenka. Optimizing modular machining design problem. In *Information Control Problems in Manufacturing (INCOM06 Symposium) de l'IFAC*. Elsevier, 2006.

[BDG+04] S. Belmokhtar, A. Dolgui, N. Guschinsky, I. Ihnatsenka, and G. Levin. Optimization of transfer lines with constraint programming approach. In *Proceedings of the 34^{th} International Conference on Computers and Industrial Engineering*, San Francisco, 14-16 Nov, 2004.

[BDGL05a] S. Belmokhtar, A. Dolgui, N. Guschinsky, and G. Levin. Un programme linéaire pour la conception de lignes de transfert à partir d'un ensemble fixe de têtes d'usinage. *Contributions sélectionnées du congrès de la ROADEF, Presses Universitaires François Rabelais*, pages 259–276, 2005.

[BDGL05b] S. Belmokhtar, A. Dolgui, N. Guschinsky, and G. Levin. A new model for transfer line with simultaneous activation of multi-spindle heads at workstations. In *Proceedings of the 35th International Conference on Computers and Industrial Engineering*, http ://www.umoncton.ca/cie/, Turkey, 19-22 Juin, 2005.

[BDGL06] S. Belmokhtar, A. Dolgui, N. Guschinsky, and G. Levin. Integer programming models for logical layout design of modular machining lines. *Computers and Industrial Engineering*, 51 :502–518, 2006.

[Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, 1957.

[BLPN03] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling*: *Applying Constraint Programming to Scheduling Problems*. Kluwer Academic Publisher, London, 2003.

[BR03] J. Bukchin and J. Rubinovitz. A weighted approach for assembly line design with station paralleling and equipment selection. *IIE Transactions*, 35 :73–85, 2003.

[Bru93] A.M Bruckstein. Why the ant trails look so staight and nice. *The Mathematical Intelligence*, 15(2) :59–62, 1993.

[BS78] K.R. Baker and L.E. Scharge. Finding an Optimal Sequence by Dynamic Programming: An Extension to Precedence-Related Tasks. *Operations Research*, 26 :111–120, 1978.

[BSW73] G.M. Buxey, N.D. Slack, and R. Wild. Production flow line system design-a review. *AIIE Transactions*, 5 :37–48, 1973.

[BT00] J. Bukchin and M. Tzur. Design of flexible assembly line to minimize equipement cost. *IIE Transactions*, 32 :585–598, 2000.

[Car82] J. Carlier. The one-machine sequencing problem. *European Journal of Operational Research*, 11 :42–47, 1982.

[Chi98] W.C. Chiang. The application of a tabu search metaheuristic to the assembly line balancing problem. *Operations Research*, 11 :209–227, 1998.

[Col02] Alain Colmerauer. *Cours de DEA sur la programmation logique*. http ://www.lim.univ-mrs.fr/, 2002.

[Dan51] G. Dantzig. *Maximization of a linear function of variables subject to linear inequalities. In*: *Activity Analysis of Production and Allocation.* Wiley, New-York, 1951.

[Dan66] S. Dano. *Industrial Production Model*. Vienna, Springer, 1966.

[Das03] A. Dashchenko. *Manufacturing Technologies for Machines of the Future* 21st *Century Technologies*. Springer, 2003.

[Dav91] L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, 1991.

[DDX89] Y. Dallery, R. David, and X. Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, 34 :943–953, 1989.

[DES06] A. Dolgui, A.V. Eremeev, and V.S. Sigaev. Hybrid algorithm for buffer allocation in tandem production lines HBBA. *Journal of Intelligent Manufacturing*, 2006. (In Press).

[DFG+05] A. Dolgui, B. Finel, N. Guschinsky, L. Levin, and F. Vernadat. An heuristic approach for transfer lines balancing. *Journal of Intelligent Manufacturing*, 16(2) :159–171, 2005.

[DFG+06a] A. Dolgui, B. Finel, O. Guschinskaya, N. Guschinsky, L. Levin, and F. Vernadat. Balancing large-scale machining lines with multi-spindle heads using decomposition. *International Journal of Production Research*, 2006. (in Press).

[DFG+06b] A. Dolgui, B. Finel, N. Guschinsky, L. Levin, and F. Vernadat. MIP approach to balancing transfer lines with blocks of parallel operations. *IIE Transactions*, 38 :869–882, 2006.

[DG89] Y. Dallery and S.B. Gershwin. Manufacturing flow line systems: a review of models and analytical results,. *Queueing Systems Theory and Applications, Special Issue on Queueing Models of Manufacturing Systems*, 12 :3–94, 1989.

[DG97] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1 :53–66, 1997.

[DGL+04] A. Dolgui, N. Guschinsky, G. Levin, M. Louly, and S. Belmokhtar. Balancing of transfer lines with simultaneously activated spindles. In *Information Control Problem in Manufacturing*: *A Proceedings Volume of the IFAC Symposium.* Elsevier, April 5-7 2004.

[DGL06] A. Dolgui, N. Gushinsky, and G. Levin. A special case of transfer lines balancing by graph approach. *European Journal of Operational Research*, 168 :732–746, 2006.

[DI05] A. Dolgui and I. Ihnatsenka. Balancing transfer lines with mixed activation of spindle heads. Technical Report 500-002, Ecole des Mines de St Etienne, 2005.

[DIB05] A. Dolgui, I. Ignatsenko, and S. Belmokhtar. Machining lines with multi-spindle stations: a new optimisation problem. *The International Journal of Ingenium*, 2 :153–162, 2005.

[DM67] E.V. Denardo and L.G. Mitten. Elements of sequential decision processes. *Journal of Industrial Engineering*, 1967.

[DMM97] M. Dell'Amico, F. Maffioli, and S. Martello. *Annotated bibliographies in combinatorial optimization.* Wiley and Sons, 1997.

[DP06] A. Dolgui and J.M. Proth. *Les systèmes de production modernes.* Lavoisier, 2006.

[ES98] E. Erel and S.C. Sarin. A survey of the assembly line balancing procedures. *Production Planning and Control*, 9(5) :414–434, 1998.

[Fal93] E. Falkenauer. The grouping genetic algorithms: widening the scope of gas. *JORBEL-Belgian Journal of Operations Research, Statistics and Computer Sciences*, 33(1-2) :79–102, 1993.

[Fal96] E. Falkenauer. A hybrid grouping algorithm for bin packing. *Journal of heuristics*, 2 :5–30, 1996.

[Fin04] B. Finel. *Structuration de lignes d'usinage : méthodes exactes et heuristiques*. PhD thesis, Université de Metz, 2004.

[GG89] S Ghosh and R.J. Gagnon. A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 4 :637–670, 1989.

[GHR88] S.C. Graves and C. Holmes Redfield. Equipment selection and task assignment for multi-product assembly system design. *International Journal of Flexible Manufacturing Systems*, 1 :31–50, 1988.

[GJ79] M.R. Garey and D.S. Johnson. *Computers and intractability: A guide to the theory of NP-Completeness*. Bell Laboratories Murray Hill, New Jersey, 1979.

[GL83] S.C. Graves and B.W. Lamar. An integer programming procedure for assembly design problems. *Operations Research*, 31(3) :522–545, 1983.

[GL97] F. Glover and M. Laguna. *Tabu Search*. Kluwer academic publishers, Boston, 1997.

[GN64] A.L. Gutjhar and G.L. Nemhauser. An algorithm for the line balancing problem. *Management Science*, 11 :308–315, 1964.

[Gol89] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, Addison Wesley, 1989.

[GP78] W.V. Gehrlein and J.H. Patterson. Balancing single model assembly lines: comments on a paper by E.M. Dar-El (Mansoor). *AIIE Transactions*, 10 :109–112, 1978.

[HB61] W.B. Helgeson and D.P. Birnie. Assembly line balancing using the ranking positional weight technique. *Journal of Industrial Engineering*, 12 :394–394, 1961.

[Hit96] K. Hitomi. *Manufacturing system engineering*. Taylor&Francis, 1996.

[HKS63] M. Held, R.M. Karp, and R. Shareshian. Assembly line balancing dynamic programming with precedence constraints. *Operations Research*, 11 :442–459, 1963.

[HM78] S.W. Haider and C.L. Moodie. An investiagtion of the use of an interactive computer model for balancing paced assembly lines. *Computers and Industrial Engineering*, 2 :83–89, 1978.

[Hof63] T.R. Hoffman. Assembly line balancing with a precedence matrix. *Management Science*, 9 :551–562, 1963.

[Hof92] T.R. Hoffman. A hybrid system for assembly line balancing. *Management Science*, 38 :39–47, 1992.

[HS00] W. Hopp and M.L Spearman. *Factory Physics*: *Foundations of Manufacturing Management*. Irwin/McGraw-Hill, 2000.

[ILO] ILOG. *ILOG Solver 6.0*: *User's Manuel*.

[Jac56] J.R. Jackson. A computing procedure for a line balancing problem. *Management Science*, 2 :261–271, 1956.

[Joh88] J.R. Johnson. Optimally balancing large assembly lines with FABLE. *Management Science*, 34 :240–253, 1988.

[Joh06] Ulf Johansson. Statistiques en bref : industrie, commerce et services. Technical Report KS-NP-06-010-FR-N, Communauté Européenne, 2006.

[Kar72] R.M. Karp. *Reducibility among combinatorial problems* in Complexity of Computer Computation, pages 85–103. Plenum Press, New York, 1972.

[KHJ+99] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritchow, H. Van Brussel, and A.G. Ulsoy. Reconfigurable manufacturing systems. *CIRP Annals*, 48(2) :527–598, 1999.

[KKK96] Y.K. Kim, Y.J. Kim, and Y. Kim. Genetic algorithm for assembly line balancing problem with various objectives. *Computers and Industrial Eginnering*, 30(3) :397–409, 1996.

[KL98] A. K. Kamrani and R. Logendran. *Group technology and cellular manufacturing*: *methodologies and applications*. Gordon and Breach, 1998.

[Kle63] M. Klein. On line balancing problem. *Operations Research*, 11 :274–281, 1963.

[KM72] V. Klee and G.J. Minty. *How good is the simplex algorithm*?. Academic Press, New York, 1972.

[Kow79] R. Kowalski. Algorithm = logic + control. *Communications of the ACM*, 22(7) :424–436, 1979.

[Koz92] J. Koza. *Genetic Programming*. Cambridge, Mass., MIT Press, 1992.

[KQ82] E.P.C. Kao and M. Queyranne. Dynamic programming methods for assembly line balancing. *Operations Research*, 30 :375–390, 1982.

[LRS06] S.D. Lapierre, A. Ruiz, and P. Soriano. Balancing assembly lines with tabu search. *European Journal of Operational Research*, 168(3) :826–837, 2006.

[Man67] E.M. Manssor. Improvement on Gutjhar and Nemhauser's algorithm for line balancing problem. *Management Science*, 14 :250–254, 1967.

[Mas06] S. Masood. Line balancing and simulation of an automated production transfer line. *Assembly Automation*, 26(1) :69–74, 2006.

[Min83] M. Minoux. *Programmation mathématique : théorie et algorithme (2 tomes)*. Dunod, Paris, 1983.

[MT99] A. Matta and T. Tolio. Flexible and productive manufacturing system architecture. In *15th Int. Conf. on Production Research*, Limerick, Ireland, Août 1999.

[MUK00] M.G. Mehrabi, A.G. Ulsoy, and Y. Koren. Reconfigurable manufacturing systems: key to futur manufacturing. *Journal of Intelligent Manufacturing*, 11 :403–419, 2000.

[MUKH02] M.G. Mehrabi, A.G. Ulsoy, Y. Koren, and P. Heytler. Trends and perspectives in flexible and reconfigurable manufacturing systems. *Journal of Intelligent Manufacturing*, 13 :135–146, 2002.

[Mul93] H. Mulenbein. *Evolutionary Algorithms*: *Theory and Applications*. Wiley, New York, 1993.

[NW98] G. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Publication, 1998.

[OL96] I.H. Osman and G. Laporte. Metaheuristics: a bibliography. *Annals of Operations Research*, 63 :513–623, 1996.

[Pas05] V. Paschos. *Optimisation combinatoire 1, concepts fondamentaux.* Hermes Science Publushing, 2005.

[PCI06] PCI. Site web, 2006. http ://www.pci.fr.

[PCPV03] H. Pierreval, C. Caux, J.L. Paris, and F. Viguier. Evolutionary approaches to the design and organization of manufacturing systems. *Computers and Industrial Engineering*, 44 :339–364, 2003.

[PDK83] P.A. Pinto, D.G. Dannenbring, and B.M. Khumawala. Assembly line balancing with processing alternatives: an application. *Management Science*, 29 :817–830, 1983.

[RDDB02] B. Rekiek, A. Dolgui, A. Dechambre, and A. Bratcu. State of art of assembly lines design optimisation. *Annual Reviews in Control*, 26(2) :163–174, 2002.

[Sak84] M. Sakarovitch. *Optimisation combinatoire, méthodes mathématiques et algorithmiques.* Hermann, 1984.

[SB87] M.J. Saltzman and I. Baybars. A two-process implicit enumeration algorithm for the simple assembly line balancing problem. *European Journal of Operational Research*, 32 :118–129, 1987.

[SB06] A. Scholl and C. Becker. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3) :666–693, 2006.

[Sch99] A. Scholl. *Balancing and Sequencing of Assembly Lines*. Physica-Verlag Heidelberg., 1999.

[SK97] A. Scholl and R. Klein. SALOME-A Bi-directional Branch and Bound Procedure for Assembly Line Balancing. *INFORMS Journal on Computing*, 9 :319–334, 1997.

[SS77] R.M. Stallman and G.J. Sussman. Forward reasonning and dependency directed backtracking in a system for computer-aided circuit analysis. *Artificial Intelligence*, 9(2) :135–196, 1977.

[Sza97] J. Szadkowski. Critical path concept for multi-tool cutting processes optimization. In *Manufacturing, Modeling, Management and Control IFAC Symposium*, pages 393–398, Vienna, Austria, 1997. Elsevier.

[Tai95] E.D Taillard. Comparaison of iterative searches for the quadratic assignment. *Location science*, 3 :87–105, 1995.

[TPG86] F.B. Talbot, J.H. Paterson, and W.V. Gehrlein. A comparative evaluation of heuristic line balancing techniques. *Management Science*, 32 :430–454, 1986.

[TYHWK03] L. Tang, D.M. Yip-Hoi, W. Wang, and Y. Koren. Concurrent line-balancing, equipement selection and throughput analysis for multi-part optimal line design. In *CIRP 2nd International Conference on Reconfigurable Manufactring (Michigan)*, 2003.

[Wik06] Wikipédia. Site web, 2006. http ://fr.wikipedia.org/wiki/Accueil.

[WM86] T.S. Wee and M.J. Magazine. Assembly Line Balancing as Generalized Bin Packing. *Operations Reasearch Letters*, 1 :56–58, 1986.

[YUA02] A. Yigit, A.G. Ulsoy, and A. Allahverdi. Optimizing modular product design for reconfigurable manufacturing. *Journal of Intelligent Manufacturing*, 13 :309–316, 2002.

[ZZX02] G.W. Zhang, S.C. Zhang, and Y. S. Xu. Research on flexible transfer line schematic design using hierarchical process planning. *Journal of Materials Processing Technology*, 129 :629–633, 2002.

École Nationale Supérieure des Mines de Saint Etienne

N° d'ordre : 424 GI

Prénom	Sana
Nom	Belmokhtar

Titre de la thèse	*Machining lines with standard equipment : modelization, configuration and optimization*
Spécialité	Industrial engineering
Mots clés	machining lines, integer programming, constraint programming

Résumé

The study investigated in this thesis proposes methods to design and optimize the configuration of machining lines from standard multi-spindle units. The problem of designing such lines is defined as the selection of a subset of spindle units from the set of all available units and their assignment to workstations such that all constraints are satisfied and the total cost is minimized. The production rate is insured by imposing a maximal cycle time to not exceed. Moreover, several constraints related to operations as precedence and inclusion constraints are taken into account. Restrictions related to spindle units are also considered such maximal number of units per station and incompatibilities between some spindles. The main part of the thesis was devoted to the design of machining line with parallel activation mode of spindle units at stations. In this case, all spindle-units of the line are activated simultaneously. Several models were proposed to solve this problem, the first one was used in a constraint programming approach for which two lower bound were suggested. Two integer linear programming models were also provided. An experimental study was done in order to compare the performances of the proposed approaches on one hand and to evaluate the different algorithms proposed to reduce the models size on the other hand. In addition, several series of tests have been achieved to study the influence of numerical characteristics of the instances on the performances of the models.

Two generalizations have been investigated : the mixed activation mode for spindle units at workstation which is a more complex problem than the parallel one, and the multi-product machining lines which considers several product at the same time.

École Nationale Supérieure des Mines de Saint Etienne

N° d'ordre : 424 GI

Prénom	Sana
Nom	Belmokhtar

Titre de la thèse	*Lignes d'usinage avec équipement standard : modélisation, configuration et optimisation*
Spécialité	Génie industriel
Mots clés	Lignes d'usinage, programmation linéaire en nombres entiers, programmation par contraintes

Résumé Les travaux de cette thèse s'inscrivent dans le cadre du développement d'outils d'aide à la décision pour la configuration des lignes d'usinage modulaires à partir d'équipements standard. Le problème de configuration se pose en termes de sélection d'un sous-ensemble d'unités d'usinage (équipements) à partir d'un ensemble connu et de leur affectation aux postes de travail définissant ainsi la structure (configuration) de la ligne. Le problème revient à trouver la meilleure solution en termes de coût de mise en oeuvre en prenant en compte différents types de contraintes : certaines sont liées à la productivité minimum à assurer, d'autres sont intrinsèques aux opérations et aux équipements. Nous nous sommes essentiellement intéressés au problème de configuration des lignes avec un mode d'activation parallèle des unités d'usinage dans les stations. Dans ce cas, le début d'un cycle est marqué par l'enclenchement simultané de toutes les unités d'usinage de la ligne. Pour ce problème, nous avons proposé trois modèles : un modèle générique que nous avons utilisé dans une approche par programmation par contraintes et deux modèles linéaires en nombres entiers. Nous avons également fourni deux bornes inférieures pour ce problème et proposé plusieurs algorithmes afin de réduire le nombre de variables des modèles et améliorer leurs performances. Une phase expérimentale a été menée pour comparer les performances des approches proposées et étudier l'impact des algorithmes de réduction. De plus, une étude de l'influence de nombreuses caractéristiques numériques des problèmes traités est également rapportée. Nous nous sommes ensuite intéressés à l'extension du plus performant de nos modèles pour deux problèmes plus généraux. La première généralisation du modèle est relative au mode d'activation des unités d'usinage où nous permettons un mode d'activation mixte et non plus seulement parallèle. La seconde généralisation porte sur le nombre de produits usinés en s'intéressant aux lignes multi-produit.